Simulação Jogo Probabilândia

Lucas dos Santos Rodrigues Szavara

2025-04-24

Import bibliotecas

```
library(dplyr)
```

```
##
## Anexando pacote: 'dplyr'
```

```
## Os seguintes objetos são mascarados por 'package:stats':
##
##     filter, lag
```

```
## Os seguintes objetos são mascarados por 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
library(SimDesign)
library(purrr)
library(tidyr)
```

Criação do Design da simulação

Nesse momento vamos considerar apenas um tabuleiro, sem as bifurcações. Também vamos considerar apenas o próximo passo, removendo situações em que ganhar menos pontos imediatamente aumenta os pontos a serem ganhos posteriormente

```
tabuleiro <- c('b', 'v', 'r', 'v', 'b', 'r', 'v', 'b', 'v', 'r', 'b', 'v', 'r',
               'b', 'v', 'r', 'b', 'v', 'r', 'r', 'r', 'b', 'v', 'b', 'r', 'r')
print(table(tabuleiro))
```

```
## tabuleiro
##  b  r  v
##  8 10  8
```

```
tabuleiro[tabuleiro == 'b'] <- 1
tabuleiro[tabuleiro == 'r'] <- 2
tabuleiro[tabuleiro == 'v'] <- 3
tabuleiro <- tabuleiro %>%
    as.double()
dados <- c(4, 6, 8, 10, 12)
```

Vamos criar uma tabela com todas as combinações a serem consideradas das variáveis a seguir:

```r
Design <- createDesign(
    # A probabilidade de acerto de uma pergunta:
    probabilidade_acerto = c(0.1, 0.3, 0.5, 0.7, 0.9),
    # Quantidade de pontos (casa branca, rosa, erro, acerto):
    pontos = list(c(3, 1, 0, 5), c(2, 1, 0, 3), c(3, 1, -5, 5)),
    # Posição atual:
    posicao = c(1, 5, 10, 15, 20, 25)
)
Design[1:5, ] %>% as.data.frame() %>% head()
```

```
##   probabilidade_acerto     pontos posicao
## 1                  0.1 3, 1, 0, 5       1
## 2                  0.3 3, 1, 0, 5       1
## 3                  0.5 3, 1, 0, 5       1
## 4                  0.7 3, 1, 0, 5       1
## 5                  0.9 3, 1, 0, 5       1
```

Simulação dos cenários

Função geradora de dados

Agora, vamos criar uma função tal que, dado um cenário, ela gera o próximo passo a ser tomado

```r
Generate <- function(condition, fixed_objects = FALSE) {
    prob_acerto <- condition$probabilidade_acerto
    # Quantidade de passos para cada dado
    passos <- dados %>%
        sapply(function(d) sample(1:d, 1))
    # Se tiver pergunta, o jogador irá acertar?
    resposta_correta <- rbinom(1, 1, prob_acerto)
    return(list(passos=passos, resposta_correta=resposta_correta))
}
```

Análise de um passo

```r
Analyse <- Analyse <- function(condition, dat, fixed_objects=FALSE) {
    pontos <- condition$pontos[[1]]
    posicao <- condition$posicao
    passos <- dat$passos
    resposta_correta <- dat$resposta_correta
    posicao_atual <- (posicao + passos) %% (1 + length(tabuleiro))
    deu_volta <- posicao + passos > length(tabuleiro)
    posicao_atual[deu_volta] <- posicao_atual[deu_volta] + 1
    tipo_posicao <- tabuleiro[posicao_atual]
    ponto <- tipo_posicao
    ponto[ponto == 3] <- ponto[ponto == 3] + resposta_correta
    return(pontos[ponto])
}
```

Agrupar analises de multiplas repetições

```r
Summarise <- function(condition, results, fixed_objects=FALSE) {
```

```r
    # Probabilidade de cada dado apresentar o melhor resultado
    p_d4 <- mean(results[, 1] == apply(results, 1, max))
    p_d6 <- mean(results[, 2] == apply(results, 1, max))
    p_d8 <- mean(results[, 3] == apply(results, 1, max))
    p_d10 <- mean(results[, 4] == apply(results, 1, max))
    p_d12 <- mean(results[, 5] == apply(results, 1, max))


    # Media de pontos obtidos ao escolher cada dado
    media_pontos_d4 <- mean(results[, 1])
    media_pontos_d6 <- mean(results[, 2])
    media_pontos_d8 <- mean(results[, 3])
    media_pontos_d10 <- mean(results[, 4])
    media_pontos_d12 <- mean(results[, 5])

    # Mediana de pontos obtidos ao escolher cada dado
    mediana_pontos_d4 <- median(results[, 1])
    mediana_pontos_d6 <- median(results[, 2])
    mediana_pontos_d8 <- median(results[, 3])
    mediana_pontos_d10 <- median(results[, 4])
    mediana_pontos_d12 <- median(results[, 5])

    # Desvio padrão de pontos obtidos ao escolher cada dado
    sd_pontos_d4 <- sd(results[, 1])
    sd_pontos_d6 <- sd(results[, 2])
    sd_pontos_d8 <- sd(results[, 3])
    sd_pontos_d10 <- sd(results[, 4])
    sd_pontos_d12 <- sd(results[, 5])
    return(c(

        p_d4 = p_d4,
        p_d6 = p_d6,
        p_d8 = p_d8,
        p_d10 = p_d10,
        p_d12 = p_d12,

        media_pontos_d4 = media_pontos_d4,
        media_pontos_d6 = media_pontos_d6,
        media_pontos_d8 = media_pontos_d8,
        media_pontos_d10 = media_pontos_d10,
        media_pontos_d12 = media_pontos_d12,

        mediana_pontos_d4 = mediana_pontos_d4,
        mediana_pontos_d6 = mediana_pontos_d6,
        mediana_pontos_d8 = mediana_pontos_d8,
        mediana_pontos_d10 = mediana_pontos_d10,
        mediana_pontos_d12 = mediana_pontos_d12,

        sd_pontos_d4 = sd_pontos_d4,
        sd_pontos_d6 = sd_pontos_d6,
        sd_pontos_d8 = sd_pontos_d8,
        sd_pontos_d10 = sd_pontos_d10,
        sd_pontos_d12 = sd_pontos_d12
```

```
    ))
}
```

Teste das funções:

Cenário escolhido:

```
condition <- Design[42, ]
print(condition$pontos[[1]])
```

```
## [1]  3  1 -5  5
```

```
print(condition$posicao)
```

```
## [1] 10
```

Resultado dos dados gerados aleatoriamente:

```
dat <- Generate(condition)
dat
```

```
## $passos
## [1] 3 4 8 5 3
##
## $resposta_correta
## [1] 1
```

Tipo das posições alcançadas usando cada dado

```
tabuleiro[condition$posicao + dat$passos]
```

```
## [1] 2 1 3 3 2
```

Pontos ganhos usando cada dado

```
results <- Analyse(condition, dat)
results
```

```
## [1] 1 3 5 5 1
```

Aparentemente, nossas funções funcionam de acordo com o esperado. Vamos rodar a simulação completa e obter os resultados agrupados

Execução da simulação

```
res <- runSimulation(
    Design,
    replications = 500,
    generate = Generate,
    analyse = Analyse,
    summarise = Summarise,
    parallel = T
)
```

```
##
## Design: 67/90;   Replications: 500   Total Time: 12.66s
##  Conditions: probabilidade_acerto=0.3, pontos=c(2, 1, 0, 3), posicao=20
##
## Design: 68/90;   Replications: 500;   RAM Used: 89.9 Mb;   Total Time: 12.97s
```

```
##  Conditions: probabilidade_acerto=0.5, pontos=c(2, 1, 0, 3), posicao=20
##
## Design: 69/90;   Replications: 500;   RAM Used: 89.9 Mb;   Total Time: 13.17s
##  Conditions: probabilidade_acerto=0.7, pontos=c(2, 1, 0, 3), posicao=20
##
## Design: 70/90;   Replications: 500;   RAM Used: 90 Mb;   Total Time: 13.43s
##  Conditions: probabilidade_acerto=0.9, pontos=c(2, 1, 0, 3), posicao=20
##
## Design: 71/90;   Replications: 500;   RAM Used: 90 Mb;   Total Time: 13.61s
##  Conditions: probabilidade_acerto=0.1, pontos=c(3, 1, -5, 5), posicao=20
##
## Design: 72/90;   Replications: 500;   RAM Used: 90 Mb;   Total Time: 13.89s
##  Conditions: probabilidade_acerto=0.3, pontos=c(3, 1, -5, 5), posicao=20
##
## Design: 73/90;   Replications: 500;   RAM Used: 90 Mb;   Total Time: 14.06s
##  Conditions: probabilidade_acerto=0.5, pontos=c(3, 1, -5, 5), posicao=20
##
## Design: 74/90;   Replications: 500;   RAM Used: 90 Mb;   Total Time: 14.27s
##  Conditions: probabilidade_acerto=0.7, pontos=c(3, 1, -5, 5), posicao=20
##
## Design: 75/90;   Replications: 500;   RAM Used: 90.1 Mb;   Total Time: 14.42s
##  Conditions: probabilidade_acerto=0.9, pontos=c(3, 1, -5, 5), posicao=20
##
## Design: 76/90;   Replications: 500;   RAM Used: 90.1 Mb;   Total Time: 14.64s
##  Conditions: probabilidade_acerto=0.1, pontos=c(3, 1, 0, 5), posicao=25
##
## Design: 77/90;   Replications: 500;   RAM Used: 90.1 Mb;   Total Time: 14.81s
##  Conditions: probabilidade_acerto=0.3, pontos=c(3, 1, 0, 5), posicao=25
##
## Design: 78/90;   Replications: 500;   RAM Used: 90.1 Mb;   Total Time: 15.00s
##  Conditions: probabilidade_acerto=0.5, pontos=c(3, 1, 0, 5), posicao=25
##
## Design: 79/90;   Replications: 500;   RAM Used: 90.2 Mb;   Total Time: 15.16s
##  Conditions: probabilidade_acerto=0.7, pontos=c(3, 1, 0, 5), posicao=25
##
## Design: 80/90;   Replications: 500;   RAM Used: 90.2 Mb;   Total Time: 15.33s
##  Conditions: probabilidade_acerto=0.9, pontos=c(3, 1, 0, 5), posicao=25
##
## Design: 81/90;   Replications: 500;   RAM Used: 90.2 Mb;   Total Time: 15.56s
##  Conditions: probabilidade_acerto=0.1, pontos=c(2, 1, 0, 3), posicao=25
##
## Design: 82/90;   Replications: 500;   RAM Used: 90.2 Mb;   Total Time: 15.73s
##  Conditions: probabilidade_acerto=0.3, pontos=c(2, 1, 0, 3), posicao=25
##
## Design: 83/90;   Replications: 500;   RAM Used: 90.2 Mb;   Total Time: 15.89s
##  Conditions: probabilidade_acerto=0.5, pontos=c(2, 1, 0, 3), posicao=25
##
## Design: 84/90;   Replications: 500;   RAM Used: 90.3 Mb;   Total Time: 16.06s
##  Conditions: probabilidade_acerto=0.7, pontos=c(2, 1, 0, 3), posicao=25
##
## Design: 85/90;   Replications: 500;   RAM Used: 90.3 Mb;   Total Time: 16.23s
##  Conditions: probabilidade_acerto=0.9, pontos=c(2, 1, 0, 3), posicao=25
##
## Design: 86/90;   Replications: 500;   RAM Used: 90.3 Mb;   Total Time: 16.38s
```

```
##  Conditions: probabilidade_acerto=0.1, pontos=c(3, 1, -5, 5), posicao=25
##
## Design: 87/90;   Replications: 500;   RAM Used: 90.3 Mb;   Total Time: 16.55s
##  Conditions: probabilidade_acerto=0.3, pontos=c(3, 1, -5, 5), posicao=25
##
## Design: 88/90;   Replications: 500;   RAM Used: 90.4 Mb;   Total Time: 16.73s
##  Conditions: probabilidade_acerto=0.5, pontos=c(3, 1, -5, 5), posicao=25
##
## Design: 89/90;   Replications: 500;   RAM Used: 90.4 Mb;   Total Time: 16.89s
##  Conditions: probabilidade_acerto=0.7, pontos=c(3, 1, -5, 5), posicao=25
##
## Design: 90/90;   Replications: 500;   RAM Used: 90.4 Mb;   Total Time: 17.08s
##  Conditions: probabilidade_acerto=0.9, pontos=c(3, 1, -5, 5), posicao=25
##
```

```
saveRDS(res, 'resultados.rds')
```

Resultados

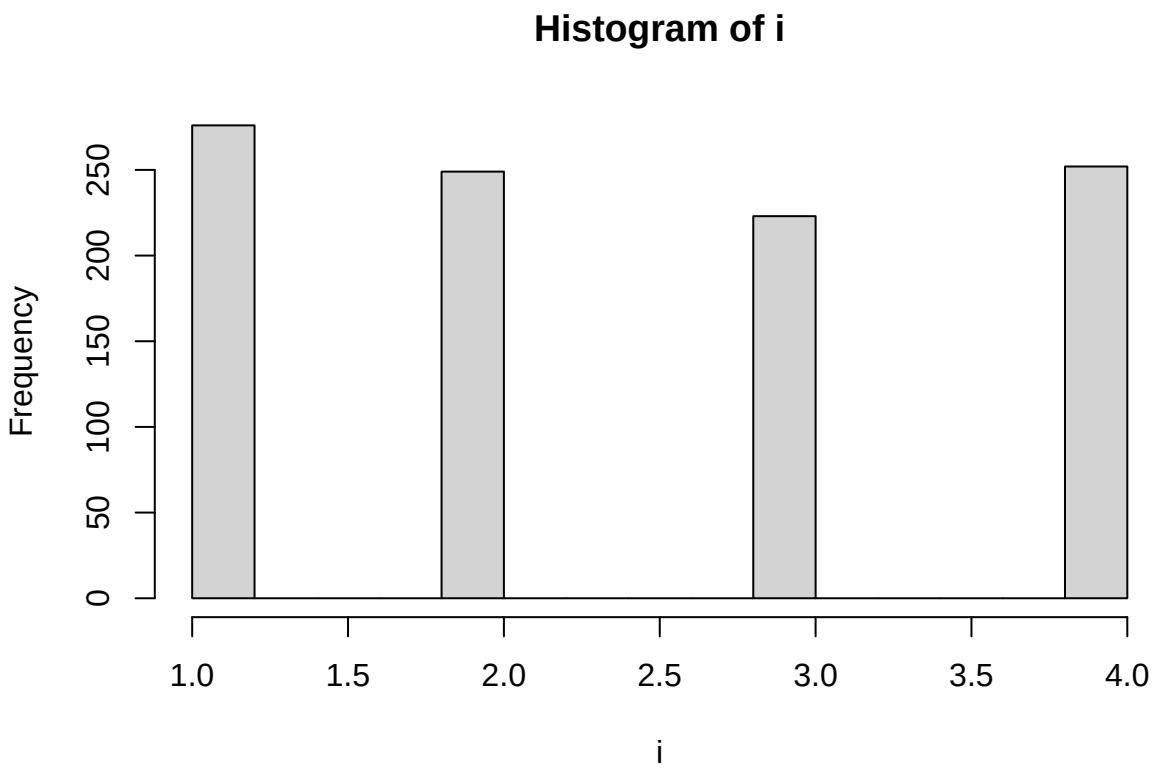
Podemos agora analisar o resultado no cenário de interesse:

```
resultado_cenario_1 <- res %>%
    mutate(pontos = as.character(pontos)) %>%
    filter(posicao == 1) %>%
    filter(probabilidade_acerto == 0.5) %>%
    filter(pontos == 'c(3, 1, 0, 5)')
resultado_cenario_1 %>%
    pivot_longer(
        cols = p_d4:sd_pontos_d12,
        names_to = c('Medida', 'Dado'),
        names_pattern = '(.*)_d(.*)'
    ) %>%
    pivot_wider(
        names_from = 'Dado',
        names_prefix = 'd'
    ) %>%
    select(Medida:d12)
```

```
## # A tibble: 4 x 6
##   Medida            d4    d6    d8   d10   d12
##   <chr>          <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 p              0.448  0.39 0.404 0.424 0.386
## 2 media_pontos   2.24   2.03 2.12  2.18  2.01
## 3 mediana_pontos 3      1    1     2     1
## 4 sd_pontos      1.87   1.97 1.87  1.76  1.78
```

Rascunho

```
i <- c()
for (j in 1:1000) {
    i <- c(i, sample(1:4, 1))
}
hist(i)
```




```
condition <- Design[68, ]
print(condition$pontos[[1]])
```

```
## [1] 2 1 0 3
```

```
print(condition$posicao)
```

```
## [1] 20
```

```
dat <- Generate(condition)
dat
```

```
## $passos
## [1]  1  6  5  2 11
##
## $resposta_correta
## [1] 1
```

Com esses tamanhos de passos, caímos na casa 23, 22, 2, 22 e 4

```
tabuleiro[c(23, 22, 2, 22, 4)]
```

```
## [1] 3 1 3 1 3
```

E, consequentemente, como erramos a pergunta, ganhamos, 3, 2, 3, 2, 3 pontos

```
results <- Analyse(condition, dat)
results
```

```
## [1] 1 1 1 2 2
```