Introdução à Ciência de Dados com R - Lista 1

Aluno: João Victor Mendes Freire

RA: 758943

Exercício 1

```
calculate_distances <- function(x1, y1, x2, y2) {
    sqrt((x2-x1)^2 + (y2-y1)^2)
}

# exemplo:
x1 = c(0, 1, 2, 3)
y1 = c(0, 1, 2, 3)

x2 = c(0, 0, 0, 0)
y2 = c(0, 0, 0, 0)

calculate_distances(x1, y1, x2, y2)
```

Exercício 2

```
minimun_coins_recursive <- function(amount, coins) {
  # Caso base, se acabaram as moedas, retorna vetor vazio
  if (length(coins) == 0) return( c() )

  # Calcula a quantia máxima da moeda atual
  curr_coin = coins[1]
  n <- amount %/% curr_coin
  amount_left <- amount %% curr_coin
  curr_quantity <- c(n)

  # Recursão
  if (length(coins) != 1)
    remaining_quantities <- minimun_coins_recursive(amount_left, coins[
2:length(coins) ])
  else
    remaining_quantities <- minimun_coins_recursive(amount_left, c())

  # Combina a lista resultante da recursão com o valor atual
  final_list <- append(curr_quantity, remaining_quantities)
  return(final_list)
}

minimun_coins <- function(amount) {
  if (amount != round(amount)) {
    print("Erro, parametro não é valor inteiro")
```

```
    return()
  }

  quantities <- minimun_coins_recursive(amount, c(100, 50, 10, 5, 2, 1))
  named_quantities <- list("100" = quantities[1][1],
                           "50" = quantities[2][1],
                           "10" = quantities[3][1],
                            "5" = quantities[4][1],
                            "2" = quantities[5][1],
                            "1" = quantities[6][1])
  return(named_quantities)
}

# Exemplo:
minimun_coins(103)
minimun_coins(4)
minimun_coins(27)
```

Exercício 3

```
validate_sudoku <- function(board) {
  validate_dimensions <- function(board) {
    return(
      nrow(board) == 9 &&
      ncol(board) == 9
    )
  }

  validate_data_type <- function(board) {
    for (row in 1:nrow(board)) {
      for (col in 1:ncol(board)) {
        element <- board[row,col]
        if ((!is.numeric(element)) || element < 1 || element > 9)
          return(FALSE)
      }
    }
    return(TRUE)
  }

  validate_rows <- function(board) {
    for (i in 1:9) {
      frequency <- rep(0, times = 9)
      for (j in 1:9) {
        frequency[board[i,j]] <- frequency[board[i,j]] + 1
        if (frequency[ board[i,j] ] > 1) {
          return(FALSE)
        }
      }
    }
    return(TRUE)
```

```
  }

  validate_cols <- function(board) {
    for (j in 1:9) {
      frequency <- rep(0, times = 9)
      for (i in 1:9) {
        frequency[board[i,j]] <- frequency[board[i,j]] + 1
        if (frequency[ board[i,j] ] > 1) {
          return(FALSE)
        }
      }
    }
    return(TRUE)
  }

  validate_sections <- function(board) {
    for (offset in 0:2) {
      for (i in (1+3*offset):(3+3*offset)) {
        frequency <- rep(0, times = 9)
        for (offset2 in 0:2) {
          for (j in (1+3*offset2):(3+3*offset2)) {
            frequency[ board[i,j] ] <- frequency[ board[i,j] ] + 1
            if (frequency[ board[i,j] ] > 1) {
              return(FALSE)
            }
          }
        }
      }
    }
    return(TRUE)
  }

  return(
    validate_dimensions(board) &&
    validate_data_type(board) &&
    validate_rows(board) &&
    validate_cols(board) &&
    validate_sections(board)
  )
}
```

Exercício 4

```
solve_sudoku <- function(board) {
  find_missing_position <- function(board) {
    for (row in 1:nrow(board)) {
      for (col in 1:ncol(board)) {
        element <- board[row,col]
        if (element == -1)
          return(c(row, col))
```

```
      }
    }
    return(NULL)
  }

  find_missing_number <- function(board, row, col) {
    # find in row
    frequency <- rep(0, times = 9)
    for (j in 1:9) {
      frequency[board[row,j]] <- frequency[board[row,j]] + 1
    }
    missing_in_row <- -1
    for (i in 1:9) {
      if (frequency[i] == 0) {
        missing_in_row <- i
        break
      }
    }

    # find in col
    frequency <- rep(0, times = 9)
    for (i in 1:9) {
      frequency[board[i,col]] <- frequency[board[i,col]] + 1
    }
    missing_in_col <- -1
    for (i in 1:9) {
      if (frequency[i] == 0) {
        missing_in_col <- i
        break
      }
    }

    if (missing_in_row != missing_in_col)
      return(-1)
    return(missing_in_col)
  }

  pos <- find_missing_position(board)
  if (is.null(pos)) {
    missing_number <- find_missing_number(board, pos[1], pos[2])
    board[pos] <- missing_number
  }
  return(board)
}
```

Exercício 5

```
# Exercício 5
library(tidyverse)
movies <- read.csv("/Users/joaovicmendes/git/aciepe-r-trabalho/Lista
```

```
1/imdb_top_1000.csv")

# a) filmes do gênero Drama
drama_movies <- filter(movies, str_detect(Genre, 'Drama'))

# b) filmes estrelados por Morgan Freeman ordenados por ano de lançamento
freeman_movies <- filter(movies, str_detect(paste0(Star1, Star2, Star3,
Star4), 'Morgan Freeman')) %>%
  arrange(Released_Year)

# c) média dos filmes dirigidos por Christpher Nolan
nolan_movies_average <- filter(movies, Director == 'Christopher Nolan')
%>%
  summarise("Média" = mean(IMDB_Rating))

# d) 5 melhores filmes (IMDB_Rating) estrelador por Brad Pitt
pitt_top_5_movies <- filter(movies, str_detect(paste0(Star1, Star2, Star3,
Star4), 'Brad Pitt')) %>%
  arrange(desc(IMDB_Rating)) %>%
  head(5)

# e) ano de lançamento médio e número médio de votos dos filmes de Ação
action_movies <- filter(movies, str_detect(Genre, 'Action')) %>%
  summarise("Ano de Lançamento Médio" = mean(as.numeric(Released_Year)),
"Média de votos" = mean(No_of_Votes))

# f) gráfico de linha com a quantidade de filmes de Aventura que foram
lançados por ano
movies$Released_Year <- as.numeric(movies$Released_Year)
adventure_movies_count <- filter(movies, str_detect(Genre, 'Adventure'))
%>%
  arrange(Released_Year) %>%
  count(Released_Year) %>%
  filter(!is.na(Released_Year))

ggplot(data = adventure_movies_count) +
  geom_line(mapping = aes(x = `Released_Year`, y = `n`)) +
  scale_x_continuous(n.breaks = 10, na.value = 0) +
  scale_y_continuous(n.breaks = 10) +
  labs(x = "Ano", y = "Quantidade") +
  theme(axis.title = element_text(size=10), plot.title =
element_text(size=12, face="bold"))

# g) gráfico de barras com o número de votos que cada filme da franquia do
Star Wars recebeu
starwars_movies <- filter(movies, str_detect(Series_Title, 'Star Wars'))
%>%
  arrange(Released_Year)
  select(Series_Title, No_of_Votes)

ggplot(data = starwars_movies) +
  geom_bar(stat = "identity", position = position_dodge(), mapping = aes(x
= `Series_Title`, y = `No_of_Votes`)) +
  labs(x = "Filme", y = "Votos") +
```

```
  theme(axis.title = element_text(size=10), plot.title =
element_text(size=12, face="bold"))
```