

Advent of Code 2021

Puzzle 1

```
puzzle1 <- read.delim('puzzle1_data.txt', header = FALSE)
puzzle1 <- as.vector(t(puzzle1))
puzzle1Shifted <- append(puzzle1[-1], 0)

length(which((puzzle1Shifted - puzzle1) > 0))

## [1] 1665

puzzle1ShiftedTwice <- append(puzzle1Shifted[-1], 0)
window3Sum <- puzzle1 + puzzle1Shifted + puzzle1ShiftedTwice
window3Sum <- head(window3Sum, -2)
window3SumShifted <- append(window3Sum[-1], 0)

length(which((window3SumShifted - window3Sum) > 0))

## [1] 1702
```

Puzzle 2

```
library(tidyr)
library(dplyr)

puzzle2 <- read.delim('puzzle2_data.txt', header = FALSE)
puzzle2 <- { puzzle2 %>%
    separate("V1", c("Direction", "Value"), sep = " ")}

puzzle2$Value <- as.numeric(puzzle2$Value)

puzzle2Grouped <- {puzzle2 %>%
    group_by(Direction) %>%
    summarise(total = sum(Value))
}

depth <- as.vector(t(puzzle2Grouped[puzzle2Grouped$Direction == "down", 2])) -
    as.vector(t(puzzle2Grouped[puzzle2Grouped$Direction == "up", 2]))

horizontal <- as.vector(t(puzzle2Grouped[puzzle2Grouped$Direction == "forward", 2]))

depth*horizontal
```

```

## [1] 2120749

puzzle2["Aim"] <- 0
puzzle2["Depth"] <- 0

for (row in 2:nrow(puzzle2)){
  puzzle2[row, "Aim"] <- ifelse(puzzle2[row, "Direction"] == "down",
                                  puzzle2[row, "Value"] + puzzle2[row-1, "Aim"],
                                  ifelse(puzzle2[row, "Direction"] == "up",
                                         puzzle2[row-1, "Aim"] - puzzle2[row, "Value"],
                                         puzzle2[row-1, "Aim"]))
  
  puzzle2[row, "Depth"] <- ifelse(puzzle2[row, "Direction"] == "forward",
                                   puzzle2[row, "Value"] * puzzle2[row, "Aim"] +
                                   puzzle2[row-1, "Depth"], puzzle2[row-1, "Depth"])
}

horizontal * puzzle2[nrow(puzzle2), "Depth"]

```

```
## [1] 2138382217
```

Puzzle 3

```

library(DescTools)

puzzle3 <- read.table('puzzle3_data.txt', header = FALSE, colClasses = "character")
puzzle3 <- { puzzle3 %>%
  separate("V1", as.character(c(1:13)), sep = "")}

puzzle3 <- puzzle3[-1]

puzzle3[] <- sapply(puzzle3, as.numeric)
modes <- puzzle3 %>%
  summarise_all(Mode)

modes <- as.vector(t(modes))
least_common <- rep(1, 12) - modes

gamma <- base::strtoi(paste(modes, collapse = ""), base = 2)
epsilon <- base::strtoi(paste(least_common, collapse = ""), base = 2)

gamma * epsilon

```

```
## [1] 3901196
```

```

puzzle3Updated <- puzzle3

for (ind in 1:ncol(puzzle3Updated)){
  currentMode <- as.vector(t(puzzle3Updated[ind] %>% summarise_all(Mode)))
  if(length(currentMode) == 1 && !is.na(currentMode[1])){

```

```

    puzzle3Updated <- filter(puzzle3Updated, puzzle3Updated[ind] == currentMode[1])
} else{
    puzzle3Updated <- filter(puzzle3Updated, puzzle3Updated[ind] == "1")
}
}

oxGen <- base::strtoi(paste(as.vector(t(puzzle3Updated))), collapse = ""), base = 2)

puzzle3UpdatedAgain <- puzzle3

for (ind in 1:ncol(puzzle3Updated)){
  if(nrow(unique(puzzle3UpdatedAgain[ind])) != 1) {
    currentMode <- as.vector(t(puzzle3UpdatedAgain[ind] %>% summarise_all_Mode()))
    if(length(currentMode) == 1 && !is.na(currentMode[1])){
      puzzle3UpdatedAgain <- filter(
        puzzle3UpdatedAgain,
        puzzle3UpdatedAgain[ind] == (1 - currentMode[1]))
    } else {
      puzzle3UpdatedAgain <- filter(puzzle3UpdatedAgain, puzzle3UpdatedAgain[ind] == "0")
    }
  }
}

co2Scrub <- base::strtoi(paste(as.vector(
  t(puzzle3UpdatedAgain)), collapse = ""),
                           base = 2)

oxGen * co2Scrub

## [1] 4412188

```

Puzzle 4

```

bingoInput <- c(13,47,64,52,60,69,80,85,57,1,2,6,30,81,86,40,27,26,97,
               77,70,92,43,94,8,78,3,88,93,17,55,49,32,59,51,28,33,
               41,83,67,11,91,53,36,96,7,34,79,98,72,39,56,31,75,82,
               62,99,66,29,58,9,50,54,12,45,68,4,46,38,21,24,18,44,
               48,16,61,19,0,90,35,65,37,73,20,22,89,42,23,15,87,74,
               10,71,25,14,76,84,5,63,95)

boards <- scan('puzzle4_data.txt')
boards <- array(boards, dim = c(5,5,100))
boards <- aperm(boards, c(2,1,3), resize = F)

naRowCol <- function(mdimArray){
  naRow <- which(apply(mdimArray, c(1,3),
                        function(x) all(is.na(x))), arr.ind = T)
  naCol <- which(apply(mdimArray, c(2,3),
                        function(x) all(is.na(x))), arr.ind = T)
  return(rbind(naRow, naCol))
}

```

```

bingo <- function(input, mdimArray){
  for (num in input){
    mdimArray[which(mdimArray == num, arr.ind = TRUE)] <- NA

    anyNaRowCol <- naRowCol(mdimArray)
    if(length(anyNaRowCol) != 0){
      return(list("slice" = anyNaRowCol[,2],
                 "lastInput" = num,
                 "updatedBoard" = mdimArray,
                 "updatedInput" = input[(which(input == num) +1) : length(input)]))
      break
    }
  }
}

firstBingo <- bingo(bingoInput, boards)

updatedBoards <- firstBingo$updatedBoard
print(sum(updatedBoards[,, firstBingo$slice], na.rm = T) *
      firstBingo$lastInput)

## [1] 49686

ind <- 100
currentInput <- bingoInput
currentBoards <- boards

while(ind > 0){
  nextBingo <- bingo(currentInput, currentBoards)
  ind <- ind - length(unique(nextBingo$slice))

  if (ind > 0){
    currentBoards <- nextBingo$updatedBoard
    for (value in nextBingo$slice){
      currentBoards[,, nextBingo$slice] <- matrix(rep(-1, 25), nrow = 5)
    }
    currentInput <- nextBingo$updatedInput
  } else {
    print(sum(nextBingo$updatedBoard[,, nextBingo$slice], na.rm = T) *
          nextBingo$lastInput)
  }
}

## [1] 26878

```

Puzzle 5

```

library(dplyr)
library(tidyr)
library(stringr)
coords <- read.delim('puzzle5_data.txt', header = FALSE)
colnames(coords) <- c("Current")
coordsSeparated <- separate(coords, "Current", c("Start", "End"), sep = "-> ")
coordsSeparated <- separate(coordsSeparated, "Start", c("Start1", "Start2"), sep = ",")
coordsSeparated <- separate(coordsSeparated, "End", c("End1", "End2"), sep = ",")
coordsSeparated <- coordsSeparated %>%
  mutate(across(where(is.character), str_trim))

coordCount <- function(fourVec, considerDiag = FALSE){
  firstCoord <- fourVec[1:2]
  secondCoord <- fourVec[3:4]

  if (firstCoord[1] == secondCoord[1]){
    coordList <- paste(firstCoord[1],
                        firstCoord[2]:secondCoord[2],
                        sep = ",")
    return (coordList)
  } else if (firstCoord[2] == secondCoord[2]){
    coordList <- paste(firstCoord[1]:secondCoord[1],
                        firstCoord[2],
                        sep = ",")
    return (coordList)
  } else if (considerDiag == TRUE){
    coordList <- paste(firstCoord[1]:secondCoord[1],
                        firstCoord[2]:secondCoord[2],
                        sep = ",")
    return (coordList)
  }
}

fullCoordList <- apply(coordsSeparated, 1,
                       function(x) coordCount(as.vector(x)))
fullCoordVec <- unlist(fullCoordList, use.names=FALSE)

print(length(which(table(fullCoordVec) > 1)))

## [1] 5442

fullCoordListDiag <- apply(coordsSeparated, 1,
                            function(x) coordCount(as.vector(x),
                                                    considerDiag = TRUE))
fullCoordVecDiag <- unlist(fullCoordListDiag, use.names=FALSE)

print(length(which(table(fullCoordVecDiag) > 1)))

## [1] 19571

```

Puzzle 6

```
library(dplyr)
library(bit64)
input <- c(1,3,4,1,1,1,1,1,1,1,1,2,2,1,4,2,
        4,1,1,1,1,1,5,4,1,1,2,1,1,1,1,4,
        1,1,1,4,4,1,1,1,1,1,1,1,2,4,1,3,
        1,1,2,1,2,1,1,4,1,1,1,4,3,1,3,1,
        5,1,1,3,4,1,1,1,3,1,1,1,1,1,1,1,
        1,1,1,1,1,5,2,5,5,3,2,1,5,1,1,
        1,1,1,1,1,1,1,1,1,2,1,1,1,1,5,
        1,1,1,1,5,1,1,1,1,1,4,1,1,1,1,1,
        3,1,1,1,1,1,1,1,1,1,1,1,3,1,2,4,
        1,5,5,1,1,5,3,4,4,4,1,1,1,2,1,1,
        1,1,1,1,2,1,1,1,1,1,5,3,1,4,1,
        1,2,2,1,2,2,5,1,1,1,2,1,1,1,1,3,
        4,5,1,2,1,1,1,1,1,5,2,1,1,1,1,1,
        1,5,1,1,1,1,1,1,1,1,5,1,4,1,5,1,1,
        1,1,1,1,1,1,1,1,1,1,1,2,1,1,1,1,
        5,4,5,1,1,1,1,1,1,1,1,5,1,1,3,1,1,
        1,3,1,4,2,1,5,1,3,5,5,2,1,3,1,1,
        1,1,1,3,1,3,1,1,2,4,3,1,4,2,2,1,
        1,1,1,1,1,5,2,1,1,1,2)

fishData <- data.frame("StartTimer" = c(0:8),
                      "fishNorm" = rep(0,9))
)

fishData$fishNorm <- as.integer64(as.vector(table(factor(input, levels= 0:8)))))

numDays <- 0
totalNeeded <- 256

while (numDays != totalNeeded){
  fishNormCurrent <- as.vector(t(fishData["fishNorm"]))
  fishNew <- fishNormCurrent[1]
  fishNormUpdated <- c(fishNormCurrent[-1], fishNew)
  fishNormUpdated[7] <- fishNormUpdated[7] + fishNew

  fishData["fishNorm"] <- fishNormUpdated

  numDays <- numDays + 1
}

print(sum.integer64(fishData$fishNorm))

## integer64
## [1] 1746710169834
```