Advent of Code 2022

Lydia Brundisch

1 12 2022

Day 1 Calorie Counting

Puzzle input

- each row is the number of calories in one food item
- each empty row separates the items one elf carries from each other

```
rm(list=ls())
temp <- read.csv("Data day 1.csv", blank.lines.skip = FALSE)
temp <- temp[, 1]
data <- as.data.frame(matrix(data = NA, 0, 0))
elf <- 1
item <- 1
for (i in 1:length(temp)) {
  if (is.na(temp[i])) {
    elf <- elf + 1
    item <- 1
  }
  else {
    data[item, elf] <- temp[i]
    item <- item + 1
  }
}

data[, 1:6]
```

```
##      V1   V2   V3    V4    V5   V6
## 1  5879 4684 5293  1745 18680 7434
## 2  4899 6694 6742 15953  3460 7346
## 3  6777 5840 4208  3781 21833 1116
## 4  5845 2705 3218    NA    NA 4079
## 5  1303 7269 5967    NA    NA 5013
## 6  6761 2127 1617    NA    NA 4597
## 7  1814 4265 5433    NA    NA 3730
## 8  6605 3944 2938    NA    NA 3660
## 9  4715 1134 6337    NA    NA 1595
## 10 2264 2623 6694    NA    NA 4501
## 11 2789 5617 1597    NA    NA 5908
## 12   NA 7191 5727    NA    NA 4847
## 13   NA   NA 1734    NA    NA   NA
## 14   NA   NA   NA    NA    NA   NA
## 15   NA   NA   NA    NA    NA   NA
```

Problem 1

- max of calories of one elf

```
data[is.na(data)] <- 0
cals_by_elf <- apply(data, 2, sum)
max(cals_by_elf)
```

```
## [1] 67016
```

Problem 2

- sum of calories of three elves with most calories

```
cals_by_elf <- sort(cals_by_elf, decreasing = TRUE)
sum(head(cals_by_elf, 3))
```

```
## [1] 200116
```

Day 2: Rock Paper Scissors

- tournament winner: player with highest total score
- total score by player: sum of all game scores by player
- game score by player: selected shape (1 for Rock, 2 for Paper, 3 for Scissors) + outcome (0 for loss, 3 for draw, and 6 for win; normal rules: Rock > Scissors, Scissors > Paper, Paper > Rock)

Puzzle input

- column 1 is opponent action (A for Rock, B for Paper, C for Scissors)
- column 2 is player action (X for Rock, Y for Paper, Z for Scissors)
- strategy: follow action plan to not win every time, but enough in total

```
rm(list=ls())
data <- read.csv("Data day 2.csv", sep = " ", header = FALSE, col.names = c("opponent_action", "player_a
data$opponent_action <- with(data, factor(opponent_action, levels = c("A", "B", "C"), labels = c("Rock"
data$player_action <- with(data, factor(player_action, levels = c("X", "Y", "Z"), labels = c("Rock", "Pa

head(data)
```

```
##   opponent_action player_action
## 1            Rock         Paper
## 2           Paper         Paper
## 3           Paper      Scissors
## 4           Paper      Scissors
## 5            Rock         Paper
## 6        Scissors         Paper
```

```r
# rows are player actions, columns are opponent actions, elements are points for player

score <- matrix(data = c(3, 0, 6, 6, 3, 0, 0, 6, 3), nrow = 3, ncol = 3, byrow = TRUE, dimnames = list(c
for (i in 1:3) {
  score[i, ] <- score[i, ] + i
}

score
```

```
##          Rock Paper Scissors
## Rock        4     1        7
## Paper       8     5        2
## Scissors    3     9        6
```

Problem 1

- total score if you follow puzzle input strategy

```r
for (i in 1:nrow(data)) {
  data$player_score[i] <- score[data$player_action[i], data$opponent_action[i]]
}

head(data)
```

```
##   opponent_action player_action player_score
## 1            Rock         Paper            8
## 2           Paper         Paper            5
## 3           Paper      Scissors            9
## 4           Paper      Scissors            9
## 5            Rock         Paper            8
## 6        Scissors         Paper            2
```

```r
sum(data$player_score)
```

```
## [1] 13565
```

Problem 2

- column 2 is win state (X for lose, Y for draw, Z for win)

```r
data$win_state <- with(data, factor(player_action, levels = c("Rock", "Paper", "Scissors"), labels = c(
data$player_action <- NULL
data$player_score <- NULL

head(data)
```

```
##   opponent_action win_state
## 1            Rock      draw
## 2           Paper      draw
## 3           Paper       win
```

```
## 4           Paper       win
## 5            Rock      draw
## 6        Scissors      draw
```

```
# rows are opponent actions, columns are win states, elements are points for player

score <- matrix(data = c("Scissors", "Rock", "Paper", "Rock", "Paper", "Scissors", "Paper", "Scissors",
temp <- factor(score, levels = c("Rock", "Paper", "Scissors"), labels = c("1", "2", "3"))
temp <- as.numeric(temp)
score <- matrix(data = temp, nrow = 3, ncol = 3, byrow = TRUE, dimnames = list(c("lose", "draw", "win")

for (i in 1:3) {
  score[i, ] <- score[i, ] + 3 * (i - 1)
}

score
```

```
##      Rock Paper Scissors
## lose    3     1        2
## draw    4     5        6
## win     8     9        7
```

- total score if you follow puzzle input strategy

```
for (i in 1:nrow(data)) {
  data$player_score[i] <- score[data$win_state[i], data$opponent_action[i]]
}

head(data)
```

```
##   opponent_action win_state player_score
## 1            Rock      draw            4
## 2           Paper      draw            5
## 3           Paper       win            9
## 4           Paper       win            9
## 5            Rock      draw            4
## 6        Scissors      draw            6
```

```
sum(data$player_score)
```

```
## [1] 12424
```

Day 3: Rucksack Reorganization

Puzzle input

- each row is a bag that is filled with letters that refer to items
- each bag can be split in half to obtain two bag compartments
- each letter is an item type (lowercase and uppercase are different)
- each item type can be translated into a number
  - lowercase item types a through z have priorities 1 through 26.

– uppercase item types A through Z have priorities 27 through 52.

```
rm(list=ls())
data <- read.csv("Data day 3.csv", header = FALSE)
data$V2 <- data$V1

data$V1 <- substr(data$V1, 1, nchar(data$V1) / 2)
data$V2 <- substr(data$V2, nchar(data$V2) / 2 + 1, nchar(data$V2))
```

Problem 1

- for each bag, find the item type in both compartments
- translate the letter to its number ("priority") and get the total sum

```
hashmap <- data.frame("Letters" = c(letters, LETTERS), "V1" = 0, "V2" = 0)

for (i in 1:nrow(data)) {
  for (j in 1:nchar(data$V1[i])) {
    hashmap[hashmap$Letters == substr(data$V1[i], j, j), 2] <- 1
  }
  for (j in 1:nchar(data$V2[i])) {
    hashmap[hashmap$Letters == substr(data$V2[i], j, j), 3] <- 1
  }
  data$duplicate[i] <- hashmap[hashmap$V1 == 1 & hashmap$V2 == 1, 1]
  hashmap$V1 <- 0
  hashmap$V2 <- 0
}

dictionary <- data.frame("Letters" = c(letters, LETTERS), "Numbers" = 1:52)

for (i in 1:nrow(data)) {
  data$priority[i] <- dictionary[data$duplicate[i] == dictionary$Letters, 2]
}

sum(data$priority)
```

```
## [1] 7446
```

Problem 2

- each group consists of 3 consecutive rows
- find letter that appears at least once in each row of a group
- get the sum of translated numbers

```
data <- read.csv("Data day 3.csv", header = FALSE)
hashmap$V3 <- 0
duplicate <- vector()
group_counter <- 1

for (i in 1:nrow(data)) {
  for (j in 1:nchar(data$V1[i])) {
```

```
    hashmap[hashmap$Letters == substr(data$V1[i], j, j), 1 + group_counter] <- 1
  }
  group_counter <- group_counter + 1
  if (group_counter == 4) {
    duplicate[i / 3] <- hashmap[hashmap$V1 == 1 & hashmap$V2 == 1 & hashmap$V3 == 1, 1]
    group_counter <- 1
    hashmap$V1 <- 0
    hashmap$V2 <- 0
    hashmap$V3 <- 0
  }
}

for (i in 1:length(duplicate)) {
  duplicate[i] <- dictionary[duplicate[i] == dictionary$Letters, 2]
}

duplicate <- as.numeric(duplicate)

sum(duplicate)
```

```
## [1] 2646
```

Day 4: Camp Cleanup

- each cell contains an ID range
- rows are pairs
- in how many pairs does one ID ranges fully contain the other?

```
rm(list=ls())
data <- read.csv(text = gsub("-", ",", readLines("Data day 4.csv")), header = FALSE)
data$containment <- 0
for (i in 1:nrow(data)) {
  if ((data$V1[i] >= data$V3[i] & data$V2[i] <= data$V4[i]) |
      (data$V1[i] <= data$V3[i] & data$V2[i] >= data$V4[i])) {
    data$containment[i] <- 1
  }
}
sum(data$containment)
```

```
## [1] 413
```

- in how many pairs do the ID ranges overlap?

```
data$overlap <- 0
for (i in 1:nrow(data)) {
  if ((data$V1[i] <= data$V3[i] & data$V2[i] >= data$V3[i]) |
      (data$V1[i] <= data$V4[i] & data$V2[i] >= data$V4[i]) |
      (data$V3[i] <= data$V1[i] & data$V4[i] >= data$V1[i]) |
      (data$V3[i] <= data$V2[i] & data$V4[i] >= data$V2[i])) {
    data$overlap[i] <- 1
  }
```

```
}
sum(data$overlap)
```

```
## [1] 806
```