Advent of Code 2021 - Day 3 Speed Edition

Gus Lipkin ~ github.com/guslipkin/AdventOfCode2021

Today was bad. I misread the instructions, forgot a bunch of basic functions, and so much more. I couldn't get any of the base reading functions to read my input properly so I ended up with `fread` from `data.table`. I also haven't been able to make any significant speed improvements from my initial solution.

Fastest solution

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
# part 1
dt <- data.table::fread("input.txt", sep = "\n",
                        colClasses = "character", data.table = FALSE) |>
  tidyr::separate(col = "V1", paste0("V", 1:13), sep = "", convert = TRUE) |>
  select(-"V1")

getModeMin <- function(x)
  return(ifelse(mean(x) > .5, 0, 1))
getModeMax <- function(x)
  return(ifelse(mean(x) > .5, 1, 0))

strtoi(paste(apply(dt, 2, getModeMax), collapse = ""), base = 2) *
  strtoi(paste(apply(dt, 2, getModeMin), collapse = ""), base = 2)
```

```
## [1] 4174964
```

```
# part 2
dto <- dt
dtc <- dt
```

```r
for (i in 1:12) {
  if (nrow(dto) > 1)
    dto <- dto |>
      filter_at(i, all_vars(. == ifelse(mean(dto[, i]) >= .5, 1, 0)))
  if (nrow(dtc) > 1)
    dtc <- dtc |>
      filter_at(i, all_vars(. == ifelse(mean(dtc[, i]) < .5, 1, 0)))
}

strtoi(paste(as.character(dto[1,]), collapse = ""), base = 2) *
  strtoi(paste(as.character(dtc[1,]), collapse = ""), base = 2)
```

```
## [1] 4474944
```

Benchmark

```r
rbenchmark::benchmark(
  "First try" = {
    library(tidyverse)
    library(data.table)
    library(compositions)

    # part 1
    dt <- data.frame(fread("input.txt", sep = "\n",
                           colClasses = c("character")))
    dt <- tidyr::separate(dt, "V1", paste0("V", 1:13),
                          sep = "", convert = TRUE)

    get_mode <- function(x)
      return(names(sort(table(x), decreasing = T, na.last = T)[1]))
    o <- c()
    for (i in 2:13)
      o <- append(o, get_mode(dt[, i]))
    o <- unbinary(paste(o, collapse = ""))

    get_mode <- function(x)
      return(names(sort(table(x), na.last = T)[1]))
    c <- c()
    for (i in 2:13)
      c <- append(c, get_mode(dt[, i]))
    c <- unbinary(paste(c, collapse = ""))

    o * c

    # part 2
    dtM <- data.frame(fread("input.txt", sep = "\n",
                            colClasses = c("character")))
    dtM <- tidyr::separate(dtM, "V1", paste0("V", 1:13),
                           sep = "", convert = TRUE)
    dtM <- dtM %>% select(-"V1")
```

```
dt <- dtM

get_modeMax <- function(x) {
  temp <- sort(table(x), decreasing = T, na.last = T)
  ifelse(temp[1] == temp[2], return(1), names(temp[1]))
}
o <- c()
for (i in 1:12) {
  dt <- dt %>% filter_at(i, all_vars(. == get_modeMax(dt[, i])))
  if (nrow(dt) == 1) {
    oxygen <- paste(as.character(dt[1, ]), collapse = "")
    break
  }
}

dt <- dtM
get_modeMin <- function(x) {
  temp <- sort(table(x), na.last = T)
  ifelse(temp[1] == temp[2], return(0), names(temp[1]))
}
c <- c()
for (i in 1:12) {
  dt <- dt %>% filter_at(i, all_vars(. == get_modeMin(dt[, i])))
  if (nrow(dt) == 1) {
    co2 <- paste(as.character(dt[1, ]), collapse = "")
    break
  }
}

unbinary(oxygen) * unbinary(co2)
},
"Second try" = {
library(tidyverse)
library(data.table)
library(compositions)

# part 1
dt <- fread("input.txt", sep = "\n", colClasses = c("character")) %>%
  as.data.frame() %>%
  tidyr::separate("V1", paste0("V", 1:13), sep = "", convert = TRUE) %>%
  select(-"V1")

getModeMin <- function(x)
  return(names(sort(table(x), na.last = T)[1]))
getModeMax <- function(x)
  return(names(sort(table(x), decreasing = T, na.last = T)[1]))

o <- unbinary(paste(apply(dt, 2, getModeMax), collapse = ""))

c <- unbinary(paste(apply(dt, 2, getModeMin), collapse = ""))

o * c
```

```
  # part 2
  dtM <- fread("input.txt", sep = "\n", colClasses = c("character")) %>%
    as.data.frame() %>%
    tidyr::separate("V1", paste0("V", 1:13), sep = "", convert = TRUE) %>%
    select(-"V1")
  dto <- dtM
  dtc <- dtM

  getModeMin <- function(x) {
    temp <- sort(table(x), na.last = T)
    ifelse(temp[1] == temp[2], return(0), names(temp[1]))
  }
  getModeMax <- function(x) {
    temp <- sort(table(x), decreasing = T, na.last = T)
    ifelse(temp[1] == temp[2], return(1), names(temp[1]))
  }

  for (i in 1:12) {
    if(nrow(dto) > 1)
      dto <- dto %>% filter_at(i, all_vars(. == getModeMax(dto[, i])))
    if(nrow(dtc) > 1)
      dtc <- dtc %>% filter_at(i, all_vars(. == getModeMin(dtc[, i])))
    if (nrow(dto) == 1 & nrow(dtc) == 1) {
      oxygen <- paste(as.character(dto[1, ]), collapse = "")
      co2 <- paste(as.character(dtc[1, ]), collapse = "")
      break
    }
  }

  unbinary(oxygen) * unbinary(co2)
  },
"Third try" = {
  library(dplyr)
  # part 1
  dt <- data.table::fread("input.txt", sep = "\n", colClasses = "character",
                          data.table = FALSE) |>
    tidyr::separate("V1", paste0("V", 1:13), sep = "", convert = TRUE) |>
    select(-"V1")

  getModeMin <- function(x)
    return(ifelse(mean(x) > .5, 0, 1))
  getModeMax <- function(x)
    return(ifelse(mean(x) > .5, 1, 0))

  strtoi(paste(apply(dt, 2, getModeMax), collapse = ""), base = 2) *
    strtoi(paste(apply(dt, 2, getModeMin), collapse = ""), base = 2)

  # part 2
  dto <- dt
  dtc <- dt

  for (i in 1:12) {
    if(nrow(dto) > 1)
```

```
        dto <- dto |>
          filter_at(i, all_vars(. == ifelse(mean(dto[, i]) >= .5, 1, 0)))
      if(nrow(dtc) > 1)
        dtc <- dtc |>
          filter_at(i, all_vars(. == ifelse(mean(dtc[, i]) < .5, 1, 0)))
    }

    strtoi(paste(as.character(dto[1, ]), collapse = ""), base = 2) *
      strtoi(paste(as.character(dtc[1, ]), collapse = ""), base = 2)
    },
  replications = 1000, columns = c(1:5), order = "user.self")
```

```
##         test replications user.self sys.self elapsed
## 3  Third try         1000    36.137    0.622  36.770
## 2 Second try         1000    49.714    1.310  51.057
## 1  First try         1000    50.850    1.343  52.201
```