Homework 8

Seth Bonney

1

```
sum_seq_of_squares <- function(n) {
  output <- 0
  for (i in 1:n) {
    out <- output + i * i
  }
  out
}

sum_seq_of_squares(15)
```

```
[1] 225
```

```
sum_seq_of_squares(27)
```

```
[1] 729
```

2

```
#part a
library(rvest)
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
```

```
library(readr)
```

```

Attaching package: 'readr'

The following object is masked from 'package:rvest':

    guess_encoding
```

```
library(janitor)
```

```

Attaching package: 'janitor'

The following objects are masked from 'package:stats':

    chisq.test, fisher.test
```

```
library(lubridate)
```

```

Attaching package: 'lubridate'

The following objects are masked from 'package:base':

    date, intersect, setdiff, union
```

```r
library(stringr)

scrape_bomojo <- function(url) {
  page <- read_html(url)

  table <- page |>
    html_element("table") |>
    html_table() |>
    clean_names() |>
    mutate(
      gross = parse_number(gross),
      theaters = parse_number(theaters),
      total_gross = parse_number(total_gross),
      release_date = mdy(paste(release_date, str_extract(url, "\\d{4}")))
    )

  select(table, -genre, -budget, -running_time, -estimated)
}

url_2024 <- "https://www.boxofficemojo.com/year/2024/"
head(scrape_bomojo(url_2024), 10)
```

```
# A tibble: 10 x 7
    rank release              gross theaters total_gross release_date distributor
   <int> <chr>                <dbl>    <dbl>       <dbl> <date>       <chr>
 1     1 Inside Out 2       6.53e8     4440   652980194 2024-06-14   Walt Disne~
 2     2 Deadpool & Wolver~ 6.37e8     4330   636745858 2024-07-26   Walt Disne~
 3     3 Wicked             4.33e8     3888   473231120 2024-11-22   Universal ~
 4     4 Moana 2            4.04e8     4200   460405297 2024-11-27   Walt Disne~
 5     5 Despicable Me 4    3.61e8     4449   361004205 2024-07-03   Universal ~
 6     6 Beetlejuice Beetl~ 2.94e8     4575   294100435 2024-09-06   Warner Bro~
 7     7 Dune: Part Two     2.82e8     4074   282144358 2024-03-01   Warner Bro~
 8     8 Twisters           2.68e8     4170   267762265 2024-07-19   Universal ~
 9     9 Godzilla x Kong: ~ 1.96e8     3948   196350016 2024-03-29   Warner Bro~
10    10 Kung Fu Panda 4    1.94e8     4067   193590620 2024-03-08   Universal ~
```

```r
# part b
scrape_bomojo2 <- function(year) {
  url <- paste0("https://www.boxofficemojo.com/year/", year, "/")
  scrape_bomojo(url)
}
```

```
head(scrape_bomojo2(2003), 10)
```

```
# A tibble: 10 x 7
    rank release             gross theaters total_gross release_date distributor
   <int> <chr>               <dbl>    <dbl>       <dbl> <date>       <chr>
 1     1 Finding Nemo       3.40e8     3425   339714978 2003-05-30   Walt Disne~
 2     2 Pirates of the Ca~ 3.05e8     3416   305413918 2003-07-09   Walt Disne~
 3     3 The Matrix Reload~ 2.82e8     3603   281576461 2003-05-15   Warner Bro~
 4     4 The Lord of the R~ 2.49e8     3703   377027325 2003-12-17   New Line C~
 5     5 Bruce Almighty     2.43e8     3549   242829261 2003-05-23   Universal ~
 6     6 X2: X-Men United   2.15e8     3749   214949694 2003-05-02   Twentieth ~
 7     7 Elf                1.68e8     3381   173398518 2003-11-07   New Line C~
 8     8 Chicago            1.68e8     2701   170687518 2003-12-27   Miramax
 9     9 Terminator 3: Ris~ 1.50e8     3504   150371112 2003-07-02   Warner Bro~
10    10 Bad Boys II        1.39e8     3202   138608444 2003-07-18   Sony Pictu~
```

3

```
library(nycflights13)
library(dplyr)
#part a and b
filter_severe <- function(data, hours = 1) {
  data |>
    filter(is.na(arr_time) | dep_delay > hours * 60)
}

flights |> filter_severe()
```

```
# A tibble: 35,138 x 19
    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
 1  2013     1     1      811            630       101     1047            830
 2  2013     1     1      826            715        71     1136           1045
 3  2013     1     1      848           1835       853     1001           1950
 4  2013     1     1      957            733       144     1056            853
 5  2013     1     1     1114            900       134     1447           1222
 6  2013     1     1     1120            944        96     1331           1213
 7  2013     1     1     1301           1150        71     1518           1345
 8  2013     1     1     1337           1220        77     1649           1531
```

```
 9  2013     1     1     1400           1250         70     1645           1502
10  2013     1     1     1505           1310        115     1638           1431
# i 35,128 more rows
# i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
#   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
#   hour <dbl>, minute <dbl>, time_hour <dttm>
```

```
flights |> filter_severe(hours = 2)
```

```
# A tibble: 18,350 x 19
    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
 1  2013     1     1      848           1835       853     1001           1950
 2  2013     1     1      957            733       144     1056            853
 3  2013     1     1     1114            900       134     1447           1222
 4  2013     1     1     1540           1338       122     2020           1825
 5  2013     1     1     1815           1325       290     2120           1542
 6  2013     1     1     1842           1422       260     1958           1535
 7  2013     1     1     1856           1645       131     2212           2005
 8  2013     1     1     1934           1725       129     2126           1855
 9  2013     1     1     1938           1703       155     2109           1823
10  2013     1     1     1942           1705       157     2124           1830
# i 18,340 more rows
# i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
#   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
#   hour <dbl>, minute <dbl>, time_hour <dttm>
```

```
#part c
summarize_severe <- function(data) {
  data |>
    summarize(
      num_cancelled = sum(is.na(arr_time)),
      num_delayed_over_1hr = sum(dep_delay > 60, na.rm = TRUE)
    )
}
flights |> group_by(dest) |> summarize_severe()
```

```
# A tibble: 105 x 3
   dest  num_cancelled num_delayed_over_1hr
   <chr>         <int>                <int>
 1 ABQ               0                   21
```

```
 2 ACK               0                  11
 3 ALB              21                  65
 4 ANC               0                   1
 5 ATL             342                1285
 6 AUS              22                 181
 7 AVL              12                  16
 8 BDL              31                  50
 9 BGR              17                  50
10 BHM              28                  50
# i 95 more rows
```

```r
#part d
summarize_weather <- function(data, var) {
  data |>
    summarize(
      min = min({{ var }}, na.rm = TRUE),
      mean = mean({{ var }}, na.rm = TRUE),
      max = max({{ var }}, na.rm = TRUE)
    )
}
weather |> summarize_weather(temp)
```

```
# A tibble: 1 x 3
    min  mean   max
  <dbl> <dbl> <dbl>
1  10.9  55.3  100.
```

```r
weather |> summarize_weather(wind_speed)
```

```
# A tibble: 1 x 3
    min  mean   max
  <dbl> <dbl> <dbl>
1     0  10.5 1048.
```

```r
#part e
standardize_time <- function(data, var) {
  data |>
    mutate(
      decimal_time = floor({{ var }} / 100) + (({{ var }} %% 100) / 60)
    )
}
flights |> standardize_time(sched_dep_time)
```

```
# A tibble: 336,776 x 20
    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
 1  2013     1     1      517            515         2      830            819
 2  2013     1     1      533            529         4      850            830
 3  2013     1     1      542            540         2      923            850
 4  2013     1     1      544            545        -1     1004           1022
 5  2013     1     1      554            600        -6      812            837
 6  2013     1     1      554            558        -4      740            728
 7  2013     1     1      555            600        -5      913            854
 8  2013     1     1      557            600        -3      709            723
 9  2013     1     1      557            600        -3      838            846
10  2013     1     1      558            600        -2      753            745
# i 336,766 more rows
# i 12 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
#   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
#   hour <dbl>, minute <dbl>, time_hour <dttm>, decimal_time <dbl>
```

4

```r
library(dplyr)

commute <- read.csv("http://aloy.rbind.io/data/CommuteAtlanta.csv")

# number of bootstrap resamples
boots <- 1000
n <- nrow(commute)
boot_means <- numeric(boots)

for (b in 1:boots) {
  sample <- commute |> slice_sample(n = n, replace = TRUE)
  boot_means[b] <- mean(sample$Time, na.rm = TRUE)
}

# 95% percentile CI
confidence_interval <- quantile(boot_means, probs = c(0.025, 0.975), na.rm = TRUE)
mean_hat <- mean(commute$Time, na.rm = TRUE)

list(
  sample_mean = mean_hat,
```

```
  percentile_CI_95 = confidence_interval
)
```

```
$sample_mean
[1] 29.11

$percentile_CI_95
    2.5%    97.5%
27.24785 30.97110
```

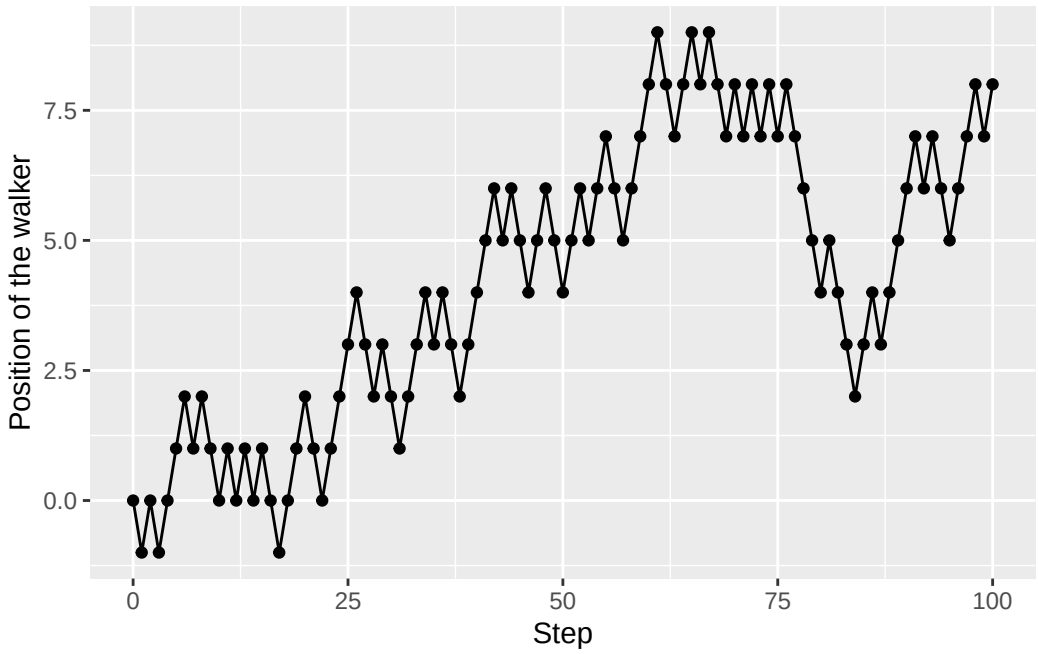
5

```
library(ggplot2)
num_of_steps <- 100
position <- numeric(num_of_steps + 1)
position[1] <- 0

for (t in 2:(num_of_steps + 1)) {
  flip <- sample(c("heads", "tails"), size = 1)
  step <- if (flip == "heads") 1 else -1
  position[t] <- position[t - 1] + step
}

# Put results in a data frame for ggplot
df_walk <- data.frame(
  Step = 0:num_of_steps,
  Position = position
)

ggplot(df_walk, aes(x = Step, y = Position)) +
  geom_line() +
  geom_point() +
  labs(
    x = "Step",
    y = "Position of the walker"
  )
```


Position of the walker
7.5
5.0
2.5
0.0
0
25
50
75
100
Step