STAT 33A Homework 5 Solutions

CJ HINES (3034590053)

Nov 5, 2020

This homework is due **Nov 5, 2020** by 11:59pm PT.

Homeworks are graded for correctness.

As you work, write your answers in this notebook. Answer questions with complete sentences, and put code in code chunks. You can make as many new code chunks as you like.

Please do not delete the exercises already in this notebook, because it may interfere with our grading tools.

You need to submit your work in two places:

- Submit this Rmd file with your edits on bCourses.
- Knit and submit the generated PDF file on Gradescope.

If you have any last-minute trouble knitting, **DON'T PANIC**. Submit your Rmd file on time and follow up in office hours or on Piazza to sort out the PDF.

The Doomsday Algorithm

Given any date, the *Doomsday algorithm* computes that date's day of the week. For example, you could use the Doomsday algorithm to compute that November 11, 1918 (the official end of World War 1) was a Monday.

The algorithm is based on the fact that in any year, several easy-to-remember dates always fall on the same day of the week. These include 4/4, 6/6, 8/8, 10/10, and 12/12. The day they fall on varies from year to year, and is called the *doomsday* for the year. For example, in 2012, the doomsday was Wednesday, so 4/4, 6/6, 8/8, 10/10, and 12/12 were all Wednesdays. In 2016, the doomsday was Monday, so the dates were all Mondays.

The doomsday algorithm has three major steps, which are explained in more detail below:

1. Compute the anchor day for the target century.
2. Compute the doomsday for the target year based on the anchor day.
3. Determine the day of week for the target date by counting the number of days to the nearest doomsday.

The Anchor Day

The *anchor day* is the doomsday for the first year in a century. The Doomsday algorithm uses the anchor day to compute the doomsday for other years in the same century.

The anchor day for a century `century` can be computed with the formula:

```
anchor = (5 * (century %% 4) + 2) %% 7
```

The result `anchor` represents the day of the week as a number from 0 (Sunday) to 6 (Saturday).

For example, suppose the target year is 1954. Then the century is 19, so you can compute the anchor day with:

```
century = 19
anchor = c
```

The result is 3, so the anchor day for 1900 to 1999 (and doomsday for 1900) is Wednesday.

Exercise 1

Write a function `compute_anchor` that accepts a (numeric) year as input and returns the anchor day (as a number) for that year's century.

Test your function for a few different years.

Hint 1: Use division and the `floor` function to isolate the century. The `floor` function rounds a number down to an integer.

Hint 2: You can use Google or Wolfram Alpha with "day of week DATE" to check your results.

YOUR ANSWER GOES HERE:

```
compute_anchor = function(year) {
  century = floor(year / 100)
  anchor = (5 * (century %% 4) + 2) %% 7
  anchor
}

compute_anchor(1999) # 3, Wednesday
```

```
## [1] 3
```

```
compute_anchor(1450) # 5, Friday
```

```
## [1] 5
```

```
compute_anchor(1789) # 0, Sunday
```

```
## [1] 0
```

The Doomsday

Once the anchor day is known, let `last2` be the last two digits of the target year. Then you can compute the doomsday for the target year with:

```
doomsday = (last2 + floor(last2 / 4) + anchor) %% 7
```

For example, suppose the target year is 1954. Then the anchor day is Wednesday, so you can compute the doomsday with:

```
anchor = 3
last2 = 54
doomsday = (last2 + floor(last2 / 4) + anchor) %% 7
```

The result is 0, so the doomsday for 1954 is Sunday.

Exercise 2

Write a function `compute_doomsday` that accepts a (numeric) year as input and returns the doomsday (as a number) for that year. Your function should call the `compute_anchor` function you wrote in Exercise 1.

Test your function for a few different years.

Hint: Use the modulo operator %% to isolate the last 2 digits of the year.

YOUR ANSWER GOES HERE:

```
compute_doomsday = function(year) {
  anchor = compute_anchor(year)
  last2 = year %% 100
  doomsday = (last2 + floor(last2 / 4) + anchor) %% 7
  doomsday
}

compute_doomsday(1954) # 0, Sunday
```

```
## [1] 0
```

```
compute_doomsday(1348) # 4, Thursday
```

```
## [1] 4
```

```
compute_doomsday(1522) # 2, Tuesday
```

```
## [1] 2
```

The Day of Week

Every month has a doomsday date (in MM/DD format here):

- Non-leap years: 1/10, 2/28
- Leap years: 1/11, 2/29
- 3/21, 4/4, 5/9, 6/6, 7/11, 8/8, 9/5, 10/10, 11/7, 12/12

The final step in the Doomsday algorithm is to compute the number of days between the target date and the doomsday date. Adding this to the doomsday, modulo 7, gives the day of the week. That is, compute:

```
(doomsday + day - doomsday_date) %% 7
```

For example, suppose the target date is July 21, 1954. The doomsday for 1954 is Sunday. The doomsday date in July is 7/11. So the day is given by:

```
doomsday = 0
day = 21
doomsday_date = 11
(doomsday + day - doomsday_date) %% 7
```

```
## [1] 3
```

In words: there are 10 days between 7/11 and 7/21. Since 7/11 is a Sunday, then 7/21 (1 week and 3 days later) must fall on a Wednesday.

Exercise 3

Write a function `compute_day` that accepts a (numeric) day, month, and year as input. Your function should return the corresponding day of week (as a string such as `"Thu"` rather than a number). Be careful of leap years!

Test your function for a few different dates.

Hint 1: First write a function `is_leap` that returns `TRUE` if a year is a leap year, and `FALSE` otherwise.

Hint 2: Use a vector of day names and subsetting to concisely and efficiently convert the day from a number (0 to 6) to a string (`"Sun"` to `"Sat"`).

YOUR ANSWER GOES HERE:

```
is_leap = function(year) {
  if (year %% 4 == 0 && year %% 100 != 0 && year %% 400 != 0) {
    TRUE
    } else {
      FALSE
    }
}

compute_day = function(day, month, year) {
  weekdays = c("Sun", "Mon","Tue","Wed", "Thu", "Fri", "Sat")
  nonleap = c(10, 28) # 1 = January, 2 = February
  leap = c(11, 29) # 1 = January, 2 = February
  dd = c(0, 0, 21, 4, 9, 6, 11, 8, 5, 10, 7, 12) # 3 = March, 4 = April, 5 = May, 6 = June, 7 = July, 8
  doomsday = compute_doomsday(year)

  if (month <= 2) {
    if (is_leap(year) == TRUE) {
      doomsday_date = leap[month]
    } else {
      doomsday_date = nonleap[month]
    }
  } else {
    doomsday_date = dd[month]
```

```
  }

  DOOMday = (doomsday + day - doomsday_date) %% 7
  weekdays[DOOMday + 1]
}

compute_day(21, 7, 1954)
```

```
## [1] "Wed"
```

```
compute_day(25, 1, 2002)
```

```
## [1] "Fri"
```

```
compute_day(25, 12, 1429)
```

```
## [1] "Fri"
```

```
compute_day(13, 2, 2020)
```

```
## [1] "Thu"
```

Exercise 4

How many times did Friday the 13th occur in 2020?

Hint: Use `sapply` to call `compute_day` for each month.

YOUR ANSWER GOES HERE:

```
months = c(1,2,3,4,5,6,7,8,9,10,11,12)
days = sapply(months, function(month) compute_day(13, month, 2020))
sum(days == "Fri")
```

```
## [1] 2
```