HW7

Lance Ding

April 2023

Instructions

Write this homework acting as if I don't know what I asked you. For example, don't simply list question numbers for the headings. If you gave this document to someone else who didn't know the assignment, they should be able to understand what you did by reading the headings, code, and accompanying text.

Look to my HW1 and RMarkdown Organization examples for how to write good headings and organize your assignment.

This HW is worth 10 total points. It leans heavily on activities from the Adventures in R Course created by Dr. Kelly Bodwin of California Polytechnic State University.

1. Change the author and date fields in the header above to your name and the date.
2. Make sure to load any packages you may need right at the start. Do *NOT* include the `learnr` package, ever, unless you are writing an interactive Tutorial (which you won't do in this) - this will cause problems.
3. Ensure that no chunks have the `include = FALSE` or `echo = FALSE` option, as I want to be able to see *all* your code and output.
4. Brief but descriptive headings and document organization (answers under headings, text near relevant code, brief explanatory text as indicated below, etc.) (1 pt)
5. For practice with `lubridate`, find the identity of the Zodiac Killer (sadly, not Ted Cruz's father) by solving Puzzle 3 from Adventures in R. (4.5 pts)

   HINTS AND CLARIFICATIONS:

   - There are many ways to accomplish this puzzle. Some - but not all - may require explicitly formatting `Time.Spotted` as a datetime and/or a date.
   - Note the problem should read the number of the *day* is 22, not the number of the *month* is 22.
   - You should *not* need to complete Question 4 (the Iceland criterion) to complete this puzzle, and it requires fighting with time zones, so please skip it.
6. **Create a map.** It must have at least two groups/regions. Those groups/regions should be colored according to some value of a third valuable. In other words, make a map that displays some interesting differences across a geographic area.

   You can choose any geographic area you want and can find map data for in R. Examples might include U.S. states; different countries in Europe, Asia, Africa, or the Americas; and Chinese provinces. (Because I suspect many of you might be interested in mapping within China: For some reason, at least for me, using `map_data("china")` and the `ggplot()` methods we discussed produces an incorrect map with odd, piecemeal provincial borders. I do not know why. If you have the same problem, as a solution I recommend using Guangchang Yu's `chinamap` package. You install it using `remotes::install_github("GuangchuangYu/chinamap")`, then create the map data frame using the `get_map_china()` function. From there you should be able to use the techniques we learned to generate a provincial map. Here are some examples of maps he made with that package. Note he uses a slightly different technique, `geom_map()`, but `geom_polygon()` seems to work just fine, too. Let me know if you're having difficulty here, though!)

The variable you map can be *anything* that varies across the geographic area you're mapping. It cannot be from a data frame included in Tutorial 7.1, but otherwise you are free to choose whatever interests you! Note this part of the assignment does require you to find some data on your own (though it can be a data frame we've worked with in the past), import it, and link it to the map data. Linking may be a bit tricky - remember the region names will have to match *exactly*, including in capitalization. Use functions like `str_to_lower()` to help with that.

Describe in narrative text what your code is doing. I want more detail than usual - you can describe the importing and linking briefly (about 1 sentence or comment per code block), but for the map-related code itself I want you to describe it essentially line by line. Basically, prove to me you can explain, in your own words, what your code is doing and how it's working to create the map. (4.5 pts)

To submit this assignment:

Ideally, knit straight to PDF by changing `html_document` to `pdf_document` in line 5 above. This should work as long as you properly installed LaTeX in Tutorial 0.1. Otherwise:

1. Knit to HTML. An HTML document should open automatically in another RStudio window.
2. Click "Open in Browser" in that HTML document. It should open as a webpage in your default browser (e.g. Chrome).
3. Click Ctrl+P/Command+P, but instead of printing a hard copy on your printer click "Save as PDF."
4. Save and upload that document to Canvas.

——BEGIN ANSWER BELOW———

Dates and Maps

Load Required Packages

```r
# remotes::install_github("GuangchuangYu/chinamap")
pacman::p_load(tidyverse, lubridate, chinamap, sp, readxl)
suspects <- read.csv("./Datasets/suspect_times.csv")
lifeExp <- read_excel("./Datasets/lifeExpectancy2010.xls")
```

```
## New names:
## * `` -> `...2`
```

Finding the Zodiac Killer

Dr. Kelly Bodwin has set us up with a challenge - find the new "Zodiac Killer", who has killed somebody in California. To help us identify this killer, we are provided with a dataset (`suspect_times.csv`) of the people who entered the building of the murder that year, complete with the date and time they entered. Additionally, Dr. Bodwin has provided us with a few clues that we can use in conjunction with our data manipulation skills to identify the killer.

Removing Morning Entry Times

The first clue we get is that the murder wasn't committed in the morning. This means that we can eliminate everyone who entered the building in the morning. However, before we can do any manipulations, we must first convert the raw data we received (`string`) to the `datetime` format. To do so, we apply the `ymd_hms()` function on the `Time.Spotted` column. Now that we have a data type that we can work with, we keep only the suspects who enter the building after the 12th hour with `hour()`, effectively eliminating all suspects who enter in the morning.

```r
summary(suspects)
```

```
##      Name            Occupation        Time.Spotted
##  Length:214         Length:214         Length:214
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
```

```r
suspects$Time.Spotted <- ymd_hms(suspects$Time.Spotted)

suspects <- suspects %>%
  filter(hour(Time.Spotted) >= 12)
```

Removing Tuesdays and Thursdays

The second clue we get is that the room that the murder was committed in is closed on Tuesdays and Thursdays, meaning the murder couldn't have taken place on those days. To remove suspects who entered on Tuesdays and Thursdays, we utilize the `wday()` function with `filter()`. The `wday()` function corresponds Sundays to `1`, Mondays to `2`, Tuesdays to `3` and so on. To eliminate all suspects who entered on a Tuesday or a Thursday, we simply select the suspects whose entry day of the week is not `3` or `5`.

```r
suspects <- suspects %>%
  filter(!(wday(Time.Spotted) %in% c(3,5)))
```

5 Weeks within Thanksgiving, 2018

Our next clue suggests that the murder was committed within 5 weeks, or 35 days, of Thanksgiving, 2018. A quick google search tells us that Thanksgiving was on the 22nd of November in 2018. We use this date to construct a lubridate `interval` object that starts 35 days before and ends 35 days after Thanksgiving, then use the `%within%` function to select the suspects who were sighted during this period.

```r
thxgvn <- ymd("2018-11-22")
suspects <- suspects %>%
  filter(Time.Spotted %within%
           interval(thxgvn-35,thxgvn+35))
```

22 on a Birthday Card

The fourth clue is about a birthday card left at the crime scene. Although the date on the card is smudged, the number of the day of the month is shown to be 22. Additionally, it is thought that the murder took place less than two days after the killer's birthday. This means that if the card belonged to the killer, we should only keep suspects who entered the building between the 22nd and 24th of any month. We achieve this with `%in%` and `mday()`, which returns the day of the month of a given `datetime` object. After this manuipulation, we are left with only 3 suspects.

```
suspects <- suspects %>%
  filter(mday(Time.Spotted) %in% c(22:24))
suspects
```

```
##              Name  Occupation        Time.Spotted
## 1   Nostradamus  astrologer 2018-12-22 15:29:50
## 2      Ted Cruz  politician 2018-12-23 15:50:10
## 3  Donna Summer      singer 2018-12-23 17:59:40
```

Minutes Divisible by 10

The fifth and final clue is actually from the killer themselves. It states that the number of minutes between Jan 1, 1970 at midnight and when the killer enters the building is evenly divisible by 10. To utilize this clue, we must first find the number of minutes between Jan 1, 1970, midnight and the entry time of each of the suspects. We achieve this through `int_length()`, `interval()`, integer division (`%/%`) and `mutate()`. By finding the duration of the `interval` object composed of the two dates in seconds and applying integer division by 60, we are able to find the number of whole minutes that have passed since the date given by the murderer. We then use the remainder operator `%%` to find the remainder of the division by 10, and only keep suspects with no remainder. We are left with a single suspect - Ted Cruz.

```
suspects <- suspects %>%
  mutate(minutes = int_length(interval(as_date(0),
                                   Time.Spotted))%/%60) %>%
  filter(minutes%%10 == 0)
suspects
```

```
##        Name  Occupation        Time.Spotted  minutes
## 1  Ted Cruz  politician 2018-12-23 15:50:10 25759670
```

Map of China

Now we are going to create a map of China. This map will utilize the geometry from `chinamap`, and the average lifespan data from a dataframe that I pulled adn renamed from this government website as a `.xls` file (I tried `.csv` but it gave me some weird unicode errors b/c Chinese characters are weird). The extracted data describes the life expectancy of the residents of each province as of 2010. To make the following manipulations easier, we will slightly clean the dataset by getting rid of the metadata and translating* the Chinese column names to English.

*I technically didn't translate the column names that R imported - that was a line of metadata. The actual column names were on the 4th line. Instead of fixing the column names then translating them I just got rid of all irrelevant metadata and added the translated names.

After cleaning `lifeExp`, we then use a `left_join()` to join the life expectancy information to our map information stored in `china`.

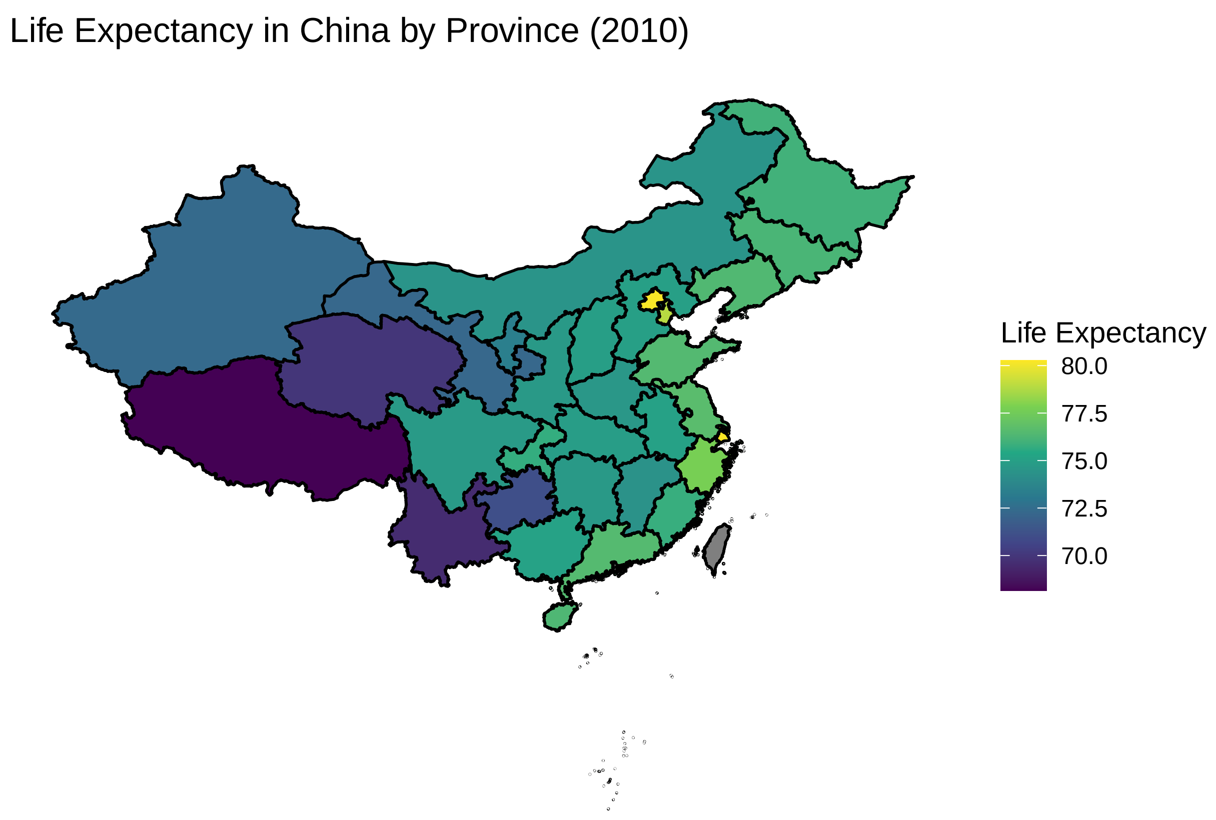
```r
colnames(lifeExp) <- c("region", "life_expectancy")
lifeExp <- lifeExp[4:34,] %>%
  mutate(life_expectancy = as.double(life_expectancy))
# head(lifeExp)

china <- get_map_china()
```

```
## Regions defined for each Polygons
```

```r
# head(china)

china_life <- china %>%
  left_join(lifeExp,
            by = c("province" = "region"))

# head(china_life)
```

Now that we have the dataframe that we want, we can construct the map. We start the map with a `ggplot()` call, passing the longitude and latitude data of each of our points as the x and y coordinates of the plot, and using our newly joined `life_expectancy` column as the fill value. We then draw the map using `geom_polygon()`, passing `aes(group = group)` to tell R to "lift its pen" when necessary, allowing us to draw the province boundaries as well as outlying islands without the lines crossing back over each other after each shape has been drawn. Finally, we throw in some color in the form of `scale_fill_viridis_c()`, clean up the unnecessary elements via the `clean_map` theme from the tutorial, and give the plot its title and labels.

```r
# I'm gonna steal your clean_map theme
clean_map <- theme(
  axis.text = element_blank(),
  axis.line = element_blank(),
  axis.ticks = element_blank(),
  panel.background = element_blank(),
  panel.border = element_blank(),
  panel.grid = element_blank(),
  axis.title = element_blank()
  )

ggplot(china_life, aes(x = long, y = lat, fill = life_expectancy)) +
  geom_polygon(aes(group = group),
               color = "black") +
  scale_fill_viridis_c(name = "Life Expectancy") +
  clean_map +
  labs(title = "Life Expectancy in China by Province (2010)")
```


Life Expectancy in China by Province (2010)
Life Expectancy
80.0
77.5
75.0
72.5
70.0