Midterm review

STAT 240

Youngjoo Yun
March 4, 2023

Goals

- To briefly review topics covered so far
  - in the context of toy data sets, and
  - with discussions on common errors.

Overview

1. Vectors in R
2. Pipes
3. ggplot2
4. dplyr
5. lubridate
6. Joining data frames
7. Filtering data frames
8. Pivoting data frames
9. Strings and regular expressions

Data sets

date	month	wday	numFlights	numPass
16.10.2023	10	Mon	5	6
24.10.2023	10	Tue	10	9
06.11.2023	11	Mon	7	7
06.11.2023	11	Mon	9	8

d1

date	month	wday	prcp
16.10.2023	10	Mon	33
25.10.2023	10	Wed	50

d2

Vectors in R

- Creating a vector using `c()`
  - `inner_join(d1, d2, by = "month")`
    `inner_join(d1, d2, by = c("month", "wday"))`
- Each column of a data frame is a vector
  - `d1$numFlights` corresponds to the vector `c(5, 10, 7, 9)`
- Boolean vectors (vectors of 0s and 1s)
  - `d1$numFlights>d1$numPass` returns `c(0, 1, 0, 1)`
  - For such vectors, `mean(d1$numFlights>d1$numPass)` corresponds to the **proportion** of entries satisfying the condition.

Also recall :, seq(),etc.

Pipes

Many functions we have come across so far take a data frame as the first argument, e.g.

1. `mutate(d1, newCol = ...)`
2. `summarize(d1, newCol = ...)`
3. `full_join(d1, d2, newCol = ...)`
4. `ggplot(df1, aes(...))`

It would be equivalent to instead write

1. `d1 %>% mutate(newCol = ...)`
2. `d1 %>% summarize(newCol = ...)`
3. `d1 %>% full_join(d2, newCol = ...)`
4. `d1 %>% ggplot(aes(...))`

`a %>% fun(b)` means "pass in a as the first argument of the function `fun()`."

Pipes: common errors

- Example 1

```
d1 %>%
 mutate(numTot = numFlights + numPass) %>%
 ggplot(d1, aes(x = numFlights, y = numPass))
```

- Example 2

```
d1 %>%
 mutate(numTot = numFlights + numPass) %>%
 group_by(wday)
 summarize(numTotFlights = sum(numFlights))
```

Pipes: common errors

- Example 1

```
d1 %>%
 mutate(numTot = numFlights + numPass) %>%
 ggplot(d1, aes(x = numFlights, y = numPass))
```

- Example 2

```
d1 %>%
 mutate(numTot = numFlights + numPass) %>%
 group_by(wday)
 summarize(numTotFlights = sum(numFlights))
```

ggplot2

Basic syntax

```
ggplot(df1, aes(x = numFlights, y = numPass)) +
 geom_xxx()
```

Use of colors

- To color differently by group

```
ggplot(df1, aes(x = numFlights, y = numPass,
 color = wday)) +
 geom_xxx()
```

- To color with a specific color

```
ggplot(df1, aes(x = numFlights, y = numPass)) +
 geom_xxx(color = "blue")
```

Also recall geom_histogram(), geom_boxplot(), geom_bar(), geom_col(), geom_point(), geom_smooth(), geom_abline(), facet_wrap(), facet_grid(), xlab(), ylab(), ggtitle(), etc.

ggplot2: common errors

- Example (contains two errors)

```
ggplot(d1, aes(x = wday, y = numFlights)) +
 geom_bar(fill = "blue") +
 scale_y_continuous(trans = "log10") +
 scale_x_continuous()
```

ggplot2: common errors

- Example (contains two errors)

```
ggplot(d1, aes(x = wday, y = numFlights)) +
 geom_bar(fill = "blue") +
 scale_y_continuous(trans = "log10") +
 scale_x_continuous()
```

dplyr

- `mutate()` is used to add a new column, change the type of an existing column, replace an existing column, etc.
- `filter()` is used to subset a data frame according to the provided conditions.
- `summarize()` is used to create a new data frame with "one row for each combination of grouping variables" (from R documentation).

grouping variables	summary statistic 1	summary statistic 2
⋮	⋮	⋮

- `group_by()` adds a group structure to the data frame.

Also recall select(), arrange(), relocate(), slice_max(), slice_min(), etc.

dplyr:group_by

`d1 %>% group_by(wday)` gives

date	month	wday	numFlights	numPass
16.10.2023	10	Mon	5	6
24.10.2023	10	Tue	10	9
06.11.2023	11	Mon	7	7
06.11.2023	11	Mon	9	8

`d1 %>% group_by(month, wday)` gives

date	month	wday	numFlights	numPass
16.10.2023	10	Mon	5	6
24.10.2023	10	Tue	10	9
06.11.2023	11	Mon	7	7
06.11.2023	11	Mon	9	8

dplyr:group_by

`d1 %>% group_by(date)` gives

date	month	wday	numFlights	numPass
16.10.2023	10	Mon	5	6
24.10.2023	10	Tue	10	9
06.11.2023	11	Mon	7	7
06.11.2023	11	Mon	9	8

`d1 %>% group_by(date, wday)` gives

date	month	wday	numFlights	numPass
16.10.2023	10	Mon	5	6
24.10.2023	10	Tue	10	9
06.11.2023	11	Mon	7	7
06.11.2023	11	Mon	9	8

dplyr:mutate() v.s. summarize()

```
d1 %>%
 group_by(wday) %>%
 mutate(TotFlights = sum(numFlights))
```

date	month	wday	numFlights	numPass	TotFlights
16.10.2023	10	Mon	5	6	21
24.10.2023	10	Tue	10	9	10
06.11.2023	11	Mon	7	7	21
06.11.2023	11	Mon	9	8	21

```
d1 %>%
 group_by(wday) %>%
 summarize(TotFlights = sum(numFlights))
```

wday	TotFlights
Mon	21
Tue	10

dplyr:common errors

- Example 1: find the average total flights by weekday.

```
d1 %>%
 group_by(date) %>%
 mutate(numFlights = sum(numFlights)) %>%
 group_by(wday) %>%
 summarize(avgFlights = mean(numFlights))
```

date	month	wday	numFlights	numPass
16.10.2023	10	Mon	5	6
24.10.2023	10	Tue	10	9
06.11.2023	11	Mon	16	7
06.11.2023	11	Mon	16	8

wday	avgFlights
Mon	$\frac{5+16+16}{3} = 12.3$
Tue	10

dplyr:common errors

- Example 1: find the average total flights by weekday.

```
d1 %>%
  group_by(date) %>%
  mutate(numFlights = sum(numFlights)) %>%
  group_by(wday) %>%
  summarize(avgFlights = mean(numFlights))
```

date	month	wday	numFlights	numPass
16.10.2023	10	Mon	5	6
24.10.2023	10	Tue	10	9
06.11.2023	11	Mon	16	7
06.11.2023	11	Mon	16	8

wday	avgFlights
Mon	$\frac{5+16+16}{3} = 12.3$
Tue	10

dplyr:common errors

- Example 2: find the average total flights by weekday.

```
d1 %>%
 group_by(date) %>%
 summarize(numFlights = sum(numFlights)) %>%
 group_by(wday) %>%
 summarize(avgFlights = mean(numFlights))
```

date	numFlights
16.10.2023	5
24.10.2023	10
06.11.2023	16

- Example 3: filter out to keep dates with flights more than 8 flights and more than 7 passengers

```
d1 %>%
 filter(numFlights > 8 | numPass > 7)
```

dplyr:common errors

- Example 2: find the average total flights by weekday.

```
d1 %>%
 group_by(date) %>%
 summarize(numFlights = sum(numFlights)) %>%
 group_by(wday) %>%
 summarize(avgFlights = mean(numFlights))
```

date	numFlights
16.10.2023	5
24.10.2023	10
06.11.2023	16

- Example 3: filter out to keep dates with flights more than 8 flights and more than 7 passengers

```
d1 %>%
 filter(numFlights > 8 | numPass > 7)
```

lubridate

- Changing to the `Date` format
  - ```
    d2 %>%
      mutate(date = dmy(date))
    ```

date	month	wday	prcp
2023-10-16	10	Mon	33
2023-10-25	10	Wed	50

- Extracting month
  - ```
    d2 %>%
      mutate(date = dmy(date)) %>%
      mutate(month = month(date, label = T))
    ```

date	month	wday	prcp
2023-10-16	Oct	Mon	33
2023-10-25	Oct	Wed	50

Also recall ymd(), mdy(), day(), yday(), year(), wday(), etc.

lubridate: common errors

- Example 1

```
d2 %>%
 mutate(month = month(date, label = T))
```

lubridate: common errors

- Example 1

```
d2 %>%
 mutate(month = month(date, label = T))
```

Joining data frames

We have studied `full_join()`, `left_join()`, `right_join()` and `inner_join()`. From now on, we work with the data frames with the `date` variable removed, i.e.,

```
d1 = d1 %>%
 mutate(date = dmy(date)) %>%
 mutate(month = month(date, label = T)) %>%
 select(-date)
d2 = d2 %>%
 mutate(date = dmy(date)) %>%
 mutate(month = month(date, label = T)) %>%
 select(-date)
```

Joining data frames

full_join

```
d1 %>%
 full_join(d2, by = c("month", "wday"))
```

month	wday	numFlights	numPass	prcp
Oct	Mon	5	6	33
Oct	Tue	10	9	NA
Nov	Mon	7	7	NA
Nov	Mon	9	8	NA
Oct	Wed	NA	NA	50

Joining data frames

left_join

```
d1 %>%
 left_join(d2, by = c("month", "wday"))
```

month	wday	numFlights	numPass	prcp
Oct	Mon	5	6	33
Oct	Tue	10	9	NA
Nov	Mon	7	7	NA
Nov	Mon	9	8	NA

Joining data frames

right_join

```
d1 %>%
 right_join(d2, by = c("month", "wday"))
```

month	wday	numFlights	numPass	prcp
Oct	Mon	5	6	33
Oct	Wed	NA	NA	50

Joining data frames

inner_join

```
d1 %>%
 inner_join(d2, by = c("month", "wday"))
```

month	wday	numFlights	numPass	prcp
Oct	Mon	5	6	33

Filtering data frames

We have studied `semi_join()` and `anti_join()`.

semi_join

```
d1 %>%
 semi_join(d2, by = c("month", "wday"))
```

month	wday	numFlights	numPass
Oct	Mon	5	6

anti_join

```
d1 %>%
 anti_join(d2, by = c("month", "wday"))
```

month	wday	numFlights	numPass
Oct	Tue	10	9
Nov	Mon	7	7
Nov	Mon	9	8

Pivoting data frames

We have studied `pivot_longer()` and `pivot_wider()`.

pivot_longer

```
d1 %>%
 pivot_longer(cols = c(numFlights, numPass),
 names_to = "whichNum", values_to = "n")
```

month	wday	whichNum	n
Oct	Mon	numFlights	5
Oct	Mon	numPass	6
Oct	Tue	numFlights	10
Oct	Tue	numPass	9
Nov	Mon	numFlights	7
Nov	Mon	numPass	7
Nov	Mon	numFlights	9
Nov	Mon	numPass	8

Pivoting data frames

pivot_wider

```
d2 %>%
 pivot_wider(names_from = wday, values_from = prcp)
```

month	Mon	Wed
10	33	50

Strings and regular expressions

Let's consider the following data frame.

word
apple
dread
forum
plain
video

d

Strings and regular expressions

- Examples of functions that don't involve regular expressions:`str_to_upper()`, `str_c()`, `str_pad()`, `str_length()`, etc.

```
d %>%
  mutate(a = str_to_upper(word),
         b = str_c(word, "123", sep = "-"),
         c = str_pad(word, 8, side = "left", pad = "-"),
         n = str_length(b))
```

word <chr>	a <chr>	b <chr>	c <chr>	n <int>
apple	APPLE	apple-123	---apple	9
dread	DREAD	dread-123	---dread	9
forum	FORUM	forum-123	---forum	9
plain	PLAIN	plain-123	---plain	9
video	VIDEO	video-123	---video	9

Strings and regular expressions

- Examples of functions that take regular expressions as arguments: `str_replace()`, `str_extract()`, `str_detect()`, etc.

Strings and regular expressions

- string: a character or characters enclosed by a pair of quotations
- regular expression: a rule "for describing patterns in strings" (from R documentation)
  - regular expression is different from a string, but **regular expression has a string representation**.
  - In a regular expression, the special characters like ^ and $, if used in their literal sense, need to be preceded by a \.
  - In the string representation of the above, we precede the \ by another \ due to \ being a special character in a string.

Strings and regular expressions

Replacing words containing a specified pattern

(Note the second argument is a string representation of a regular expression)

```
d %>%
  mutate(rc = str_replace(word, ".", "X"),
         rv = str_replace(word, "[aeiou]", "X"),
         rvs = str_replace(word, "[aeiou]+", "X"),
         rvs2 = str_replace(word, "[aeiou]{2,}", "X"),
         rvcccv = str_replace(word, "[aeiou]...[aeiou]",
         "X"))
```

word <chr>	rc <chr>	rv <chr>	rvs <chr>	rvs2 <chr>	rvcccv <chr>
apple	Xpple	Xpple	Xpple	apple	X
dread	Xread	drXad	drXd	drXd	dread
forum	Xorum	fXrum	fXrum	forum	forum
plain	Xlain	plXin	plXn	plXn	plain
video	Xideo	vXdeo	vXdeo	vidX	video

Strings and regular expressions

Group operator

```
d %>%
  mutate(a = str_detect(word, "^(.)...\\1$"),
         b = str_detect(word, "^(.)(.)..\\2$"))
```

word <chr>	a <lgl>	b <lgl>
apple	FALSE	FALSE
dread	TRUE	FALSE
forum	FALSE	FALSE
plain	FALSE	FALSE
video	FALSE	FALSE