Prep2S24

Sofia Guttmann

2024-02-08

Reminder: Prep assignments are to be completed individually. Upload a final copy of the .Qmd and renamed .pdf to your private repo, and submit the renamed pdf to Gradescope before Sunday, Feb. 11th at midnight (11:59 pm is what Gradescope shows).

Reading

The associated reading for the week is Chapter 4, Chapter 5, Chapter 6 (skip 6.4) and Sections 8.3 and 8.4. This reading explores major functions in wrangling data, including reshaping data. There are many commands here to learn about - do your best to develop a sense of what they each do, and we will build on that by using them for the rest of the semester. You don't need to memorize them all.

Remember, I recommend you code along with the book examples. You can try out the code yourself - just be sure to load the mdsr package and any other packages referenced. You can get the code in R script files (basically, files of just R code, not like a .Rmd or .Qmd) from the book website.

1 - Some basics

Many different data wrangling commands are covered in these chapters. Identify the command you'd use for each of the operations below.

> part a - Add the average of 3 variables to the data set as a new variable.

Solution: You can use the **`mutate()`** function from the **`dplyr`** package

> part b - Keep only 4 columns of a data frame in a new data set.

Solution: You can use the **`select()`** function from the **`dplyr`** package

> part c - Choose observations that match a particular category of a categorical variable to keep in a new data set.

Solution: You can use the **`filter()`** function from the **`dplyr`** package

> part d - Combine two data sets based on common variables where all rows from the first are returned, along with any matches in the second.

Solution: You can use the **`left_join()`** function from the **`dplyr`** package

2 - NYC Flights

In Section 5.1, the `flights` and `airlines` tables within the `nycflights13` package are joined together.

> part a - Recreate the `flights_joined` dataset from Section 5.1, being sure to *glimpse* the data in the Console (or via the code chunk) to verify the join worked.

Solution:

```r
library(dplyr)
library(nycflights13)

flights_joined <- inner_join(flights, airlines, by = "carrier")

print(head(flights_joined))
```

```
# A tibble: 6 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
# 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>, name <chr>
```

> part b - Now, starting from `flights_joined`, create a new dataset `flights_short` that does the following:

- creates a new variable, `distance_km`, which is distance in kilometers (note that 1 mile is about 1.6 kilometers)
- keeps only the variables: `name`, `flight`, `arr_delay`, and `distance_km` and
- keeps only observations where the distance is less than 480 kilometers (300 miles).

Solution:

```r
flights_joined <- flights_joined %>%
  mutate(distance_km = distance * 1.6)
```

```r
flights_short <- flights_joined %>%
  select(name, flight, arr_delay, distance_km) %>%
  filter(distance_km < 480)

print(head(flights_short))
```

```
# A tibble: 6 x 4
  name                     flight arr_delay distance_km
  <chr>                     <int>     <dbl>       <dbl>
1 ExpressJet Airlines Inc.   5708       -14        366.
2 JetBlue Airways            1806        -4        299.
3 Southwest Airlines Co.     4646       -19        296
4 ExpressJet Airlines Inc.   4144        12        339.
5 JetBlue Airways            1002       -10        299.
6 JetBlue Airways              20        -1        422.
```

part c - Using the functions introduced in Section 4.1.4, compute the number of flights (call this `N`), the average arrival delay (call this `avg_arr_delay`), and the average distance in kilometers (call this `avg_dist_km`) among these flights with distances less than 480 km (i.e. working off of `flights_short`), grouping by the carrier name. Sort the results in descending order based on `avg_arr_delay`. Save the results in a tibble object called `delay_summary`, and display the table.

Solution:

```r
library(dplyr)
library(tibble)


delay_summary <- flights_short %>%
  group_by(name) %>%
  summarize(N = n(),
            avg_arr_delay = mean(arr_delay, na.rm = TRUE),
            avg_dist_km = mean(distance_km, na.rm = TRUE)) %>%
  arrange(desc(avg_arr_delay))


print(delay_summary)
```

```
# A tibble: 11 x 4
   name                            N avg_arr_delay avg_dist_km
```

```
   <chr>                      <int>          <dbl>        <dbl>
 1 Mesa Airlines Inc.           319         18.0           361.
 2 ExpressJet Airlines Inc. 15588         15.6           372.
 3 Envoy Air                  2924         11.0           350.
 4 JetBlue Airways           10813          8.36          360.
 5 Endeavor Air Inc.          5779          6.84          319.
 6 Southwest Airlines Co.      208          4.92          272.
 7 United Air Lines Inc.      3353          4.09          320.
 8 SkyWest Airlines Inc.         1          3             366.
 9 US Airways Inc.            9633          2.22          309.
10 American Airlines Inc.     1455          1.88          299.
11 Delta Air Lines Inc.       1214         -0.643         325.
```

part d - Rename the four columns in the `delay_summary` data table to `Airline`, `"Total flights under 480 km"`, `"Average arrival delay (mins)"` and `"Average distance (km)"`, respectively, then use `kable(booktabs = TRUE, digits = 0)` to make the final table output in the pdf close to publication quality.

Solution:

```
library(knitr)


names(delay_summary) <- c("Airline", "Total flights under 480 km", "Average arrival delay


kable(delay_summary, booktabs = TRUE, digits = 0)
```

Airline	Total flights under 480 km	Average arrival delay (mins)	Average distance (km)
Mesa Airlines Inc.	319	18	361
ExpressJet Airlines Inc.	15588	16	372
Envoy Air	2924	11	350
JetBlue Airways	10813	8	360
Endeavor Air Inc.	5779	7	319
Southwest Airlines Co.	208	5	272
United Air Lines Inc.	3353	4	320
SkyWest Airlines Inc.	1	3	366
US Airways Inc.	9633	2	309
American Airlines Inc.	1455	2	299
Delta Air Lines Inc.	1214	-1	325

3 - Baby names - Variant of 6.2.5 example

> part a - Working with the `babynames` data in the **babynames** package, create a dataset `recent_names` that only includes years 2003 to 2017 (giving us the most recent 15 years of data).

Solution:

```
library(babynames)
library(dplyr)


recent_names <- babynames %>%
  filter(year >= 2003 & year <= 2017)

print(head(recent_names))
```

```
# A tibble: 6 x 5
   year sex   name        n    prop
  <dbl> <chr> <chr>   <int>   <dbl>
1  2003 F     Emily   25688 0.0128
2  2003 F     Emma    22704 0.0113
3  2003 F     Madison 20197 0.0101
4  2003 F     Hannah  17634 0.00879
5  2003 F     Olivia  16146 0.00805
6  2003 F     Abigail 15925 0.00794
```

> part b - Following the code presented in Section 6.2.5, create a dataset called `recentnames_summary` that summarizes the total number of people in recent history (years 2003 to 2017) with each name, grouped by sex.

Solution:

```
library(babynames)
library(dplyr)


recent_names <- babynames %>%
  filter(year >= 2003 & year <= 2017)

recentnames_summary <- recent_names %>%
  group_by(name, sex) %>%
```

```
  summarize(total_people = sum(n, na.rm = TRUE))
```

```
`summarise()` has grouped output by 'name'. You can override using the
`.groups` argument.
```

```
print(head(recentnames_summary))
```

```
# A tibble: 6 x 3
# Groups:   name [6]
  name      sex   total_people
  <chr>     <chr>        <int>
1 Aaban     M              107
2 Aabha     F               35
3 Aabid     M               10
4 Aabir     M                5
5 Aabriella F               32
6 Aada      F                5
```

part c - Now, following the third and fourth code chunks presented in Section 6.2.5, reshape or *pivot* the summary data from *long* format to *wide* format. Only keep observations where more than 8,000 babies have been named in each sex (`M` and `F`), and find the smaller of the two ratios `M / F` and `F / M` to identify the top three sex-balanced names (and only the top three!). Save the wide data as `recentnames_balanced_wide`. Display the table.

Solution:

```
recentnames_summary_wide <- recentnames_summary %>%
  pivot_wider(names_from = sex, values_from = total_people)

recentnames_summary_wide_filtered <- recentnames_summary_wide %>%
  filter(M > 8000 & F > 8000)

recentnames_summary_wide_filtered <- recentnames_summary_wide_filtered %>%
  mutate(ratio_M_F = M / F,
         ratio_F_M = F / M)
 names
```

```
function (x)  .Primitive("names")
```

```r
top_three_balanced_names <- recentnames_summary_wide_filtered %>%
  mutate(min_ratio = pmin(ratio_M_F, ratio_F_M)) %>%
  top_n(3, min_ratio)


recentnames_balanced_wide <- top_three_balanced_names


print(recentnames_balanced_wide)
```

```
# A tibble: 26 x 6
# Groups:   name [26]
   name          M      F ratio_M_F ratio_F_M min_ratio
   <chr>     <int>  <int>     <dbl>     <dbl>     <dbl>
 1 Alexis    29951 118367     0.253    3.95      0.253
 2 Amari     14612   9983     1.46     0.683     0.683
 3 Angel    125664  26344     4.77     0.210     0.210
 4 Avery     28018 102487     0.273    3.66      0.273
 5 Cameron  116636  12529     9.31     0.107     0.107
 6 Casey     12033   8278     1.45     0.688     0.688
 7 Charlie   19563  12818     1.53     0.655     0.655
 8 Dakota    23823  18971     1.26     0.796     0.796
 9 Dylan    173360   9267    18.7      0.0535    0.0535
10 Emerson   11125  17993     0.618    1.62      0.618
# i 16 more rows
```

part d - Finally, use `pivot_longer()` to put the dataset back into *long* form. Call this dataset `recentnames_balanced` and display the table.

Solution:

```r
recentnames_balanced <- recentnames_balanced_wide %>%
  pivot_longer(cols = c(M, F, ratio_M_F, ratio_F_M, min_ratio),
               names_to = "Variable",
               values_to = "Value")

print(recentnames_balanced)
```

```
# A tibble: 130 x 3
# Groups:   name [26]
   name   Variable       Value
```

```
   <chr>  <chr>          <dbl>
 1 Alexis M          29951
 2 Alexis F         118367
 3 Alexis ratio_M_F      0.253
 4 Alexis ratio_F_M      3.95
 5 Alexis min_ratio      0.253
 6 Amari  M          14612
 7 Amari  F           9983
 8 Amari  ratio_M_F      1.46
 9 Amari  ratio_F_M      0.683
10 Amari  min_ratio      0.683
# i 120 more rows
```

4 - Ethics

Each subsection of Section 8.4 discusses an ethical scenario and ends with one or more questions. Consider the subsection 8.4.6 on "Reproducible spreadsheet analysis".

Write two or three sentences reflecting on how using RMarkdown would help avoid some of the issues described in this scenario, or at least make them easier to spot.

Solution: Using RMarkdown allows for reproducibility and transparency. Using one file ensures ease when tracking changes and transformations.