Homework 06 CSCI 036

Lucas Welch

Due: Friday, 2022-10-21

Instructions

Please box your answers. For numerical answers, this can be done using something like $\boxed{34}$. For text answers, this can be done using something like $\boxed{\text{My answer}}$. The output of a code chunk is automatically boxed, so no need to do more.

Consider the following tibbles.

```
t1 <- tibble(
  emp_id = c(1001, 1116, 1239),
  class = c("temp", "full", "full")
)
t2 <- tibble(
  emp_id = c(1001, 1211, 1239, 1543),
  comp = c(16232, 42003, 51522, 44023)
)
```

a. What is the primary key in `t1`, that is, a key that consists of a single variable?

b. Write code to add the information from the `comp` variable of `t2` to the three observations in `t1`, but only with the rows of `t1`.

c. Write code to remove the observations from `t1` where there are no `comp` values in `t2`. The columns of the output should be the same as those of `t1`.

a. $\boxed{\text{emp\_id}}$

b.

```
t1 |>
  left_join(t2)
```

```
## Joining, by = "emp_id"
```

```
## # A tibble: 3 × 3
##   emp_id class  comp
##    <dbl> <chr> <dbl>
## 1   1001 temp  16232
## 2   1116 full     NA
## 3   1239 full  51522
```

c.

```
t1 |>
  semi_join(t2)
```

```
## Joining, by = "emp_id"
```

```
## # A tibble: 2 × 2
##   emp_id class
##    <dbl> <chr>
## 1   1001 temp
## 2   1239 full
```

#semi_join: It keeps only rhe values from t1 that have values in t2

Consider the nycflights13 package.

```
library(nycflights13)
```

The `airports` dataset contains the FAA designation of airports, along with their name and the latitude and longitude of the destination.

```
airports |> head()
```

```
## # A tibble: 6 × 8
##   faa   name                             lat   lon   alt    tz dst   tzone
##   <chr> <chr>                          <dbl> <dbl> <dbl> <dbl> <chr> <chr>
## 1 04G   Lansdowne Airport               41.1 -80.6  1044    -5 A     America/Ne…
## 2 06A   Moton Field Municipal Airport   32.5 -85.7   264    -6 A     America/Ch…
## 3 06C   Schaumburg Regional             42.0 -88.1   801    -6 A     America/Ch…
## 4 06N   Randall Airport                 41.4 -74.4   523    -5 A     America/Ne…
## 5 09J   Jekyll Island Airport           31.1 -81.4    11    -5 A     America/Ne…
## 6 0A9   Elizabethton Municipal Airport  36.4 -82.2  1593    -5 A     America/Ne…
```

The `flights` dataset contains data for over 300,000 flights leaving the New York metropolitan area.

```
flights |> head()
```

```
## # A tibble: 6 × 19
##    year month   day dep_time sched_dep…¹ dep_d…² arr_t…³ sched…⁴ arr_d…⁵ carrier
##   <int> <int> <int>    <int>       <int>   <dbl>   <int>   <int>   <dbl> <chr>
## 1  2013     1     1      517         515       2     830     819      11 UA
## 2  2013     1     1      533         529       4     850     830      20 UA
## 3  2013     1     1      542         540       2     923     850      33 AA
## 4  2013     1     1      544         545      -1    1004    1022     -18 B6
## 5  2013     1     1      554         600      -6     812     837     -25 DL
## 6  2013     1     1      554         558      -4     740     728      12 UA
## # … with 9 more variables: flight <int>, tailnum <chr>, origin <chr>,
## #   dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>,
## #   time_hour <dttm>, and abbreviated variable names ¹sched_dep_time,
## #   ²dep_delay, ³arr_time, ⁴sched_arr_time, ⁵arr_delay
```

Consider just keeping the origin and dest information along with a key for each flight.

```
origin_dest <-
  flights |>
  select(year:day, origin, dest)
```

Note that the name of the key in the `airport` dataset is `faa`, while the airport name might be either origin or dest. Still, a `left_join` can be used by setting the `by` parameter correctly. For instance, to join the longitude and latitude of the destination airport of the flight, use:

```
origin_dest |>
  left_join(airports, by = c("dest" = "faa")) |>
  head()
```

```
## # A tibble: 6 × 12
##    year month   day origin dest  name           lat   lon   alt    tz dst   tzone
##   <int> <int> <int> <chr>  <chr> <chr>        <dbl> <dbl> <dbl> <dbl> <chr> <chr>
## 1  2013     1     1 EWR    IAH   George Bus…   30.0 -95.3    97    -6 A     Amer…
## 2  2013     1     1 LGA    IAH   George Bus…   30.0 -95.3    97    -6 A     Amer…
## 3  2013     1     1 JFK    MIA   Miami Intl    25.8 -80.3     8    -5 A     Amer…
## 4  2013     1     1 JFK    BQN   <NA>          NA    NA      NA    NA <NA>  <NA>
## 5  2013     1     1 LGA    ATL   Hartsfield…   33.6 -84.4  1026    -5 A     Amer…
## 6  2013     1     1 EWR    ORD   Chicago Oh…   42.0 -87.9   668    -6 A     Amer…
```

Modify the above code to join the longitude and latitude of the origin airport for flights in `origin_dest` .

```
origin_dest |>
  left_join(airports, by = c("origin" = "faa")) |>
  head()
```

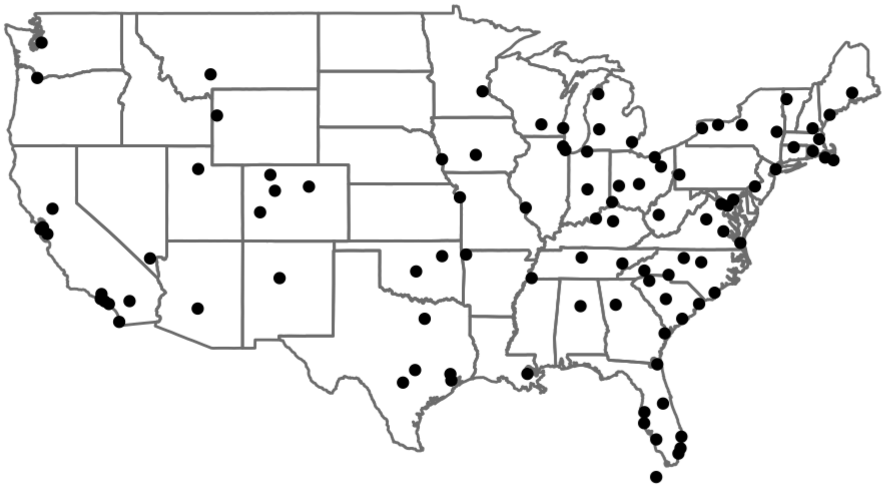
```
## # A tibble: 6 × 12
##    year month   day origin dest  name           lat   lon   alt    tz dst   tzone
##   <int> <int> <int> <chr>  <chr> <chr>        <dbl> <dbl> <dbl> <dbl> <chr> <chr>
## 1  2013     1     1 EWR    IAH   Newark Lib…   40.7 -74.2    18    -5 A     Amer…
## 2  2013     1     1 LGA    IAH   La Guardia    40.8 -73.9    22    -5 A     Amer…
## 3  2013     1     1 JFK    MIA   John F Ken…   40.6 -73.8    13    -5 A     Amer…
## 4  2013     1     1 JFK    BQN   John F Ken…   40.6 -73.8    13    -5 A     Amer…
## 5  2013     1     1 LGA    ATL   La Guardia    40.8 -73.9    22    -5 A     Amer…
## 6  2013     1     1 EWR    ORD   Newark Lib…   40.7 -74.2    18    -5 A     Amer…
```

(Exercise 13.6.1.1 of R for Data Science by Grolemund & Wickham)

a. Write code to compute the average flight arrival delay from the `flights` dataset (removing NA values) grouped by destination airport.

b. The following code draws a map of the United States with the airports that are destinations in the continental United States in the `flights` dataset.

```
flights |>
  select(dest) |>
  unique() |>
  left_join(select(airports, faa, lat, lon),
            by = c("dest" = "faa")) |>
  filter(lon > -140) |>
  ggplot(aes(lon, lat)) +
    borders("state") +
    geom_point() +
    theme_void() +
    coord_quickmap()
```



Note that by using `unique()`, each airport only appears once in the tibble being plotted, which makes the number of rows 105 instead of over 330,000. This speeds up the graphic plotting by a lot! Modify this code to color the airports in this plot based on their average flight delay.

a.

```
flights |>
  group_by(dest) |>
  mutate(on_time_flights = arr_delay >= 0 ) |>
  summarize(avg_on_time = mean(on_time_flights, na.rm = TRUE))
```

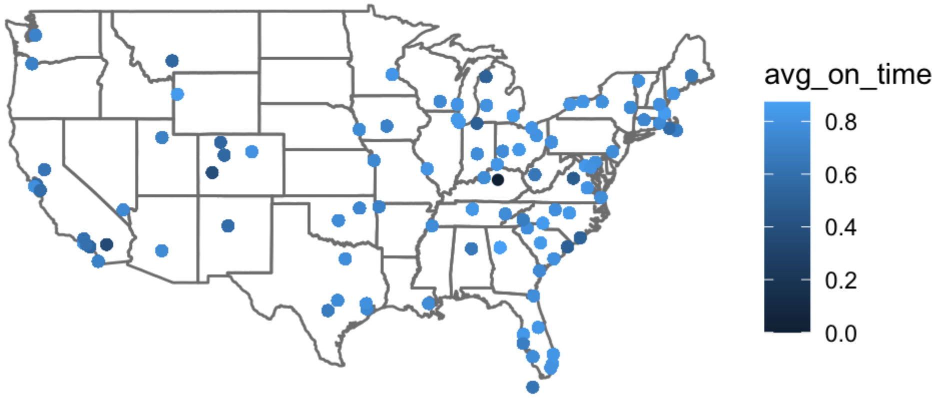
```
## # A tibble: 105 × 2
##    dest  avg_on_time
##    <chr>       <dbl>
##  1 ABQ         0.429
##  2 ACK         0.413
##  3 ALB         0.452
##  4 ANC         0.625
##  5 ATL         0.493
##  6 AUS         0.419
##  7 AVL         0.487
##  8 BDL         0.359
##  9 BGR         0.397
## 10 BHM         0.465
## # … with 95 more rows
```

b.

```
 flights |>
  select(dest, arr_delay) |>
  unique() |>
  left_join(select(airports, faa, lat, lon),
            by = c("dest" = "faa")) |>
  group_by(dest) |>
  mutate(on_time_flights = arr_delay >= 0 ) |>
  select(arr_delay, on_time_flights, lon, lat) |>
  mutate(avg_on_time = mean(on_time_flights, na.rm = TRUE)) |>

  filter(lon > -140) |>
  ggplot(aes(lon, lat, color = avg_on_time)) +
    borders("state") +
    geom_point() +
    theme_void() +
    coord_quickmap()
```

```
## Adding missing grouping variables: `dest`
```



#COME BACK REVIEW

(Exercise 13.6.1.1 of R for Data Science by Grolemund & Wickham)

Write code to add an artificial (surrogate) key to `flights` with the row_number function which is the first column of the new tibble.

```
flights |>
  mutate(key = row_number(), .before = 1)
```

```
## # A tibble: 336,776 × 20
##      key  year month   day dep_time sched_dep_…¹ dep_d…² arr_t…³ sched…⁴ arr_d…⁵
##    <int> <int> <int> <int>    <int>        <int>   <dbl>   <int>   <int>   <dbl>
##  1     1  2013     1     1      517          515       2     830     819      11
##  2     2  2013     1     1      533          529       4     850     830      20
##  3     3  2013     1     1      542          540       2     923     850      33
##  4     4  2013     1     1      544          545      -1    1004    1022     -18
##  5     5  2013     1     1      554          600      -6     812     837     -25
##  6     6  2013     1     1      554          558      -4     740     728      12
##  7     7  2013     1     1      555          600      -5     913     854      19
##  8     8  2013     1     1      557          600      -3     709     723     -14
##  9     9  2013     1     1      557          600      -3     838     846      -8
## 10    10  2013     1     1      558          600      -2     753     745       8
## # … with 336,766 more rows, 10 more variables: carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>, and abbreviated variable names
## #   ¹sched_dep_time, ²dep_delay, ³arr_time, ⁴sched_arr_time, ⁵arr_delay
```

(Exercise 13.6.1.1 of R for Data Science by Grolemund & Wickham modified.)

a. Create a tibble that counts the number of times that any particular plane has flown–removing those observations where the tail number is unknown. Filter this tibble to only include planes that have flown at least 100 times.

b. Using a semi-join, filter `flights` to only show flights with planes that have flown at least 100 flights.

a.

```
TB <- flights |>
  filter(!is.na(tailnum)) |>
  group_by(tailnum) |>
  summarize(count = n())|>
  filter(count >= 100)
TB
```

```
## # A tibble: 1,217 × 2
##    tailnum count
##    <chr>   <int>
##  1 N0EGMQ    371
##  2 N10156    153
##  3 N10575    289
##  4 N11106    129
##  5 N11107    148
##  6 N11109    148
##  7 N11113    138
##  8 N11119    148
##  9 N11121    154
## 10 N11127    124
## # … with 1,207 more rows
```

b.

```
flights |>
  semi_join(TB)
```

```
## Joining, by = "tailnum"
## # A tibble: 228,390 × 19
##     year month   day dep_time sched_de…¹ dep_d…² arr_t…³ sched…⁴ arr_d…⁵ carrier
##    <int> <int> <int>    <int>      <int>   <dbl>   <int>   <int>   <dbl> <chr>
##  1  2013     1     1      517        515       2     830     819      11 UA
##  2  2013     1     1      533        529       4     850     830      20 UA
##  3  2013     1     1      544        545      -1    1004    1022     -18 B6
##  4  2013     1     1      554        558      -4     740     728      12 UA
##  5  2013     1     1      555        600      -5     913     854      19 B6
##  6  2013     1     1      557        600      -3     709     723     -14 EV
##  7  2013     1     1      557        600      -3     838     846      -8 B6
##  8  2013     1     1      558        600      -2     849     851      -2 B6
##  9  2013     1     1      558        600      -2     853     856      -3 B6
## 10  2013     1     1      558        600      -2     923     937     -14 UA
## # … with 228,380 more rows, 9 more variables: flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>, and abbreviated variable names
## #   ¹sched_dep_time, ²dep_delay, ³arr_time, ⁴sched_arr_time, ⁵arr_delay
```

State which of the following

- `pivot_wider`
- `pivot_longer`
- `separate`
- `unite`

is the right function to use in the situation.

a. Two variable values need to be combined into one variable entry.

b. The entries of a column should be column names instead.

c. An entry contains values for two or more variables.

d. The column names are measurements rather than something you measure.

a. unite

b. pivot_wider

c. separate

d. pivot_longer

Consider the following dataset that gives the number of days of rainfall for five cities over three months.

```
df1 <- read_csv2('City;January;February;March
  Atlanta, Georgia;11;10;10
  Austin, Texas;7;7;9
  Baltimore, Maryland;10;9;10
  Birmingham, Alabama;11;10;10
  Boston, Massachusetts;11;10;12')
df1
```

```
## # A tibble: 5 × 4
##   City                   January February March
##   <chr>                    <dbl>    <dbl> <dbl>
## 1 Atlanta, Georgia            11       10    10
## 2 Austin, Texas                7        7     9
## 3 Baltimore, Maryland         10        9    10
## 4 Birmingham, Alabama         11       10    10
## 5 Boston, Massachusetts       11       10    12
```

a. Tidy this data.

b. Find the mean number of days of rainfall from January through March for each of the five cities.

a.

```
df1 |>
  separate(City, into = c("city", "states"))
```

```
## # A tibble: 5 × 5
##   city       states        January February March
##   <chr>      <chr>           <dbl>    <dbl> <dbl>
## 1 Atlanta    Georgia            11       10    10
## 2 Austin     Texas               7        7     9
## 3 Baltimore  Maryland           10        9    10
## 4 Birmingham Alabama            11       10    10
## 5 Boston     Massachusetts      11       10    12
```

b.

```
df1 |>
  pivot_longer('January':'February':'March', names_to = "month", values_to = "rain_value
s") |>
  group_by(City) |>
  summarize(total_rain_fall = mean(rain_values))
```

```
## Warning in x:y: numerical expression has 2 elements: only the first used
## # A tibble: 5 × 2
##   City                  total_rain_fall
##   <chr>                           <dbl>
## 1 Atlanta, Georgia                10.3 
## 2 Austin, Texas                    7.67
## 3 Baltimore, Maryland              9.67
## 4 Birmingham, Alabama             10.3 
## 5 Boston, Massachusetts           11   
```

Now suppose the data from the last problem was presented in the following CSV file:

```
df2 <- read_csv2("
  Atlanta, Georgia;Jan/11
  Atlanta, Georgia;Feb/10
  Atlanta, Georgia;Mar/10
  Austin, Texas;Jan/7
  Austin, Texas;Feb/7
  Austin, Texas;Mar/9
  Baltimore, Maryland;Jan/10
  Baltimore, Maryland;Feb/9
  Baltimore, Maryland;Mar/10
  Birmingham, Alabama;Jan/11
  Birmingham, Alabama;Feb/10
  Birmingham, Alabama;Mar/10
  Boston, Massachusetts;Jan/11
  Boston, Massachusetts;Feb/10
  Boston, Massachusetts;Mar/12",
  col_names = c("City", "Days_rain")
)
```

a. Tidy this data.

b. Create a horizontal bar plot that for each of the three months, making five bars for the five cities where the height of each bar is the total number of days of rain for that city from January through March.

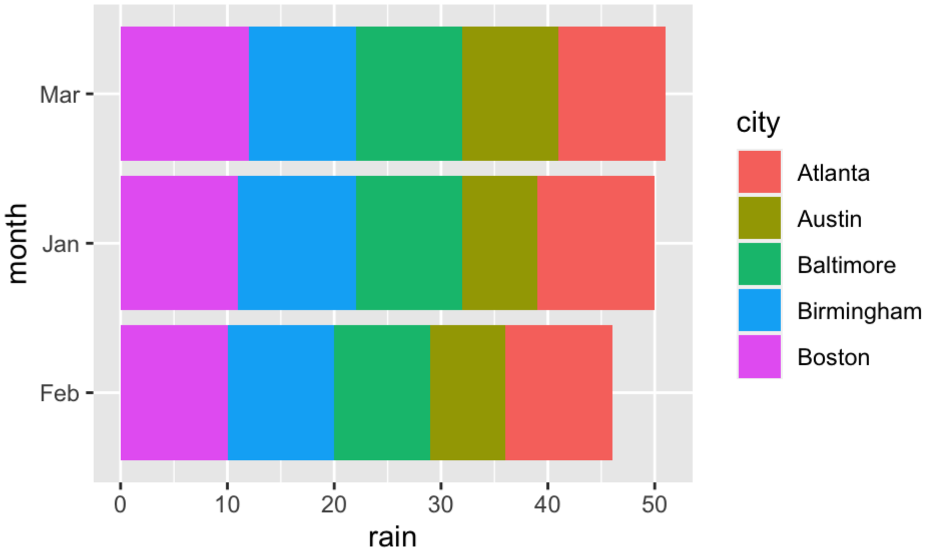
a.

```
df2 |>
  separate(City, into = c("city", "state")) |>
  separate(Days_rain, into = c("month", "rain")) |>
  pivot_wider(names_from = month, values_from = rain)
```

```
## # A tibble: 5 × 5
##   city       state         Jan   Feb   Mar
##   <chr>      <chr>         <chr> <chr> <chr>
## 1 Atlanta    Georgia       11    10    10
## 2 Austin     Texas         7     7     9
## 3 Baltimore  Maryland      10    9     10
## 4 Birmingham Alabama       11    10    10
## 5 Boston     Massachusetts 11    10    12
```

b.

```
df2 |>
  separate(City, into = c("city", "state")) |>
  separate(Days_rain, into = c("month", "rain"), convert = TRUE) |>
  ggplot(aes(x = month, y = rain)) +
  geom_bar(aes(fill = city), stat = 'identity') +
  coord_flip()
```


Mar
Jan
Feb
month
0
10
20
30
40
50
rain
city
Atlanta
Austin
Baltimore
Birmingham
Boston

Consider the following data set.

```
state_data <- tribble(
  ~state, ~type, ~number,
  "Oregon", "year_founded", 1859,
  "Oregon", "population_2020_census", 4237256,
  "Oregon", "population_2010_census", 3831074,
  "California", "year_founded", 1850,
  "California", "population_2020_census", 39538223,
  "Washington", "population_2020_census", 7705281,
  "Washington", "population_2010_census", 6724540,
)
```

a. Are all the values under `number` measuring the same thing?

b. Tidy this data.

c. How many NA values are in your table?

a. no

b.

```
state_data |>
  pivot_wider(names_from = type, values_from = number)
```

```
## # A tibble: 3 × 4
##   state      year_founded population_2020_census population_2010_census
##   <chr>             <dbl>                  <dbl>                  <dbl>
## 1 Oregon             1859                4237256                3831074
## 2 California         1850               39538223                     NA
## 3 Washington           NA                7705281                6724540
```

c. 2

Consider the following dataset. Bring together the first and last names into a single field `name`, where the last name goes first, followed by a comma, then a space, and finally the first name. For instance if the first name was Pete and the last name Agular, the result should be be `Aguilar, Pete`.

```
jan6_committee <-
  tribble(
  ~first_name, ~last_name, ~membership,
  "Bennie", "Thompson", "Chairperson",
  "Zoe", "Lofgren", "Majority",
  "Adam", "Schiff", "Majority",
  "Pete", "Aguilar", "Majority",
  "Stephanie", "Murphy", "Majority",
  "Jamie", "Raskin", "Majority",
  "Elaine", "Luria", "Majority",
  "Liz", "Cheney", "Minority",
  "Adam", "Kinzinger", "Minority"
  )
jan6_committee
```

```
## # A tibble: 9 × 3
##   first_name last_name membership
##   <chr>      <chr>     <chr>
## 1 Bennie     Thompson  Chairperson
## 2 Zoe        Lofgren   Majority
## 3 Adam       Schiff    Majority
## 4 Pete       Aguilar   Majority
## 5 Stephanie  Murphy    Majority
## 6 Jamie      Raskin    Majority
## 7 Elaine     Luria     Majority
## 8 Liz        Cheney    Minority
## 9 Adam       Kinzinger Minority
```

```
jan6_committee |>
  unite(col = full_name, last_name, first_name, sep = ",")
```

```
## # A tibble: 9 × 2
##   full_name        membership
##   <chr>            <chr>
## 1 Thompson,Bennie  Chairperson
## 2 Lofgren,Zoe      Majority
## 3 Schiff,Adam      Majority
## 4 Aguilar,Pete     Majority
## 5 Murphy,Stephanie Majority
## 6 Raskin,Jamie     Majority
## 7 Luria,Elaine     Majority
## 8 Cheney,Liz       Minority
## 9 Kinzinger,Adam   Minority
```