Richard Gresham
CPSC 375-01
October 8, 2022

Homework #5

Prepare your answers as a **single PDF file**.
Group work: You may work in groups of 1-3. Include all group member names in the PDF file. Only one person in the group should submit to Canvas.
Due: check on Canvas.

1. Consider the two tables shown below called **population** and **countyseats**.

population:

	state	county	year	population
1	California	Orange	2000	2846289
2	California	Orange	2010	3010232
3	California	Los Angeles	2000	3694820
4	California	Los Angeles	2010	3792621

countyseats:

	statename	countyname	countyseat
1	California	Orange	Santa Ana
2	California	Los Angeles	Los Angeles
3	California	San Diego	San Diego
4	Oregon	Wasco	The Dalles

You should be able to calculate the output by hand though you may use R to check your answer.
Draw the output table from the following operations (you should be able to calculate the output by hand though you may use R to check your answers).

a) `population %>% inner_join(countyseats)`

```
nothing would occur as there is no matching columns based off of the
named columns above. This is because inner_join() retains only
observations where there is a match in both datasets and in the above
example while certain column values maintain similarities, since they
are named differently their will be an error such as is observed in
Rstudtio. We can create make two different column names in datasets be
compared together using the by command as seen in b).
```

b) `population %>% inner_join(countyseats, by=c(state="statename"))`

```
state              county year   population  countyname countyseat
1  California      Orange 2000    2846289      Orange   Santa Ana
2  California      Orange 2000    2846289 Los Angeles Los Angeles
3  California      Orange 2000    2846289   San Diego   San Diego
4  California      Orange 2010    3010232      Orange   Santa Ana
```

```
5  California      Orange 2010    3010232 Los Angeles Los Angeles
6  California      Orange 2010    3010232   San Diego   San Diego
7  California Los Angeles 2000    3694820      Orange   Santa Ana
8  California Los Angeles 2000    3694820 Los Angeles Los Angeles
9  California Los Angeles 2000    3694820   San Diego   San Diego
10 California Los Angeles 2010    3792621      Orange   Santa Ana
11 California Los Angeles 2010    3792621 Los Angeles Los Angeles
12 California Los Angeles 2010    3792621   San Diego   San Diego
```

c) `population %>% inner_join(countyseats, by=c(state="statename", county="countyname"))`

```
state             county year population   countyseat
1 California      Orange 2000    2846289   Santa Ana
2 California      Orange 2010    3010232   Santa Ana
3 California Los Angeles 2000    3694820 Los Angeles
4 California Los Angeles 2010    3792621 Los Angeles
```

d) `population %>% inner_join(countyseats, by=c(state="statename", county="countyname", year="countyseat"))`

in this example there is no data set that would match this. This is because there are no matchings of year and countyseat and thus nothing to write.

2. Consider the `billboard` dataset that is supplied with the tidyverse which shows the Billboard top 100 song rankings in the year 2000. Apply the tidyverse's data wrangling verbs to answer these questions. For each question, **give only the code**.
Table becoming tidy via code below:
billboard2 <- billboard %>% pivot_longer(cols = starts_with("wk"), names_to = "week",names_prefix = "wk",names_transform = list(week = as.integer), values_to = "rank", values_drop_na = TRUE)

a) Show for each track, how many weeks it spent on the chart

   billboard2 %>% group_by(track) %>% summarize(maxweek = max(week))

b) List tracks in decreasing order of number of weeks spent on the chart

   billboard2 %>% group_by(track) %>% summarize(maxweek = max(week)) %>% arrange(desc(maxweek))

c) Show for each track, its top rank

   billboard2 %>% group_by(track) %>% summarize('TopRank' = min(rank))

d) List tracks in increasing order of its top rank billboard2 %>% group_by(track) %>% summarise('TopRank' = min(rank)) %>% arrange(TopRank)

e) Show for each artist, their top rank

   billboard2 %>% group_by(artist) %>% summarise('TopRank' = min(rank))

f) List artists in increasing order of their top rank

   billboard2 %>% group_by(artist) %>% summarise('TopRank' = min(rank)) %>% arrange(TopRank)

g) List tracks that spent more than 35 weeks in the charts.

   billboard2 %>% group_by(track) %>% summarise(Morethan35 = max(week)) %>% filter(Morethan35 > 35) %>% select(track, Morethan35)

h) List tracks that spent more than 35 weeks in the charts along with their artists
   #same as above but I'm also printing out the artist of the songs as well.

   billboard2 %>% group_by(track, artist) %>% summarise(Morethan35 = max(week)) %>% filter(Morethan35 > 35) %>% select(artist,track, Morethan35)

Hint: ***First***, **convert to a tidy table**. Show code first for this step. All the above questions can then be answered with a single data pipeline.

3. The `demographics.csv`[1] file (available in the Datasets module on Canvas) gives the proportion of a country's population in different age groups and some other demographic data such as mortality rates and expected lifetime. You can read a CSV file into a tibble using tidyverse's `read_csv()`, like so: `demo <- read_csv("demographics.csv")`

(a) The data is not "tidy". In 2-3 sentences, explain why.
    This data is not because the Series Code and the YR2015 are really messy to read. The variable name Series name is also not too helpful since its already on the Series code anyways. The table can also be grouped by country names and have multiple series code columns to solve this issue.

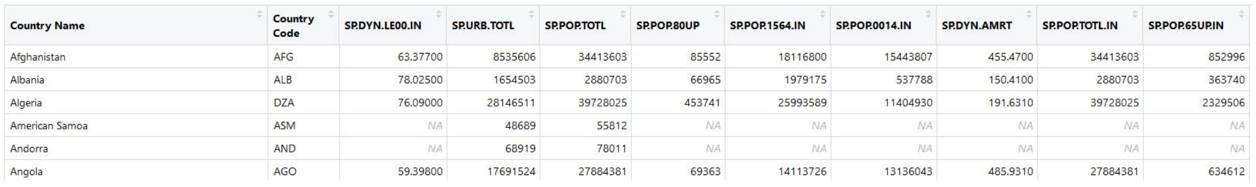
(b) Transform the table to tidy data with one country per row. [Give code]
    demo1 <- demo %>% select(-'Series Name') %>% pivot_wider(names_from = 'Series Code', values_from = YR2015)

[1] Original dataset: https://databank.worldbank.org/source/population-estimates-and-projections/Type/TABLE/preview/on#

(c) Add the male/female population numbers together (i.e., ignore sex-related differences). [Hint: You will have to mutate for every pair of columns, e.g., mutate(SP.POP.0014.IN=SP.POP.0014.MA.IN+SP.POP.0014.FE.IN] [Give code]

```
combo_demo <- demo1 %>% mutate(SP.POP.80UP = SP.POP.80UP.FE +
SP.POP.80UP.MA) %>% mutate(SP.POP.1564 = SP.POP.1564.FE.IN +
SP.POP.1564.MA.IN) %>% mutate(SP.POP.0014.IN = SP.POP.0014.FE.IN +
SP.POP.0014.MA.IN) %>% mutate(SP.DYN.AMRT = SP.DYN.AMRT.FE +
SP.DYN.AMRT.MA) %>% mutate(SP.POP.TOTL.IN = SP.POP.TOTL.FE.IN +
SP.POP.TOTL.MA.IN) %>% mutate(SP.POP.65UP.IN = SP.POP.65UP.FE.IN +
SP.POP.65UP.MA.IN) %>% select(c(`Country Name`, `Country Code`, SP.DYN.LE00.IN,
SP.URB.TOTL, SP.POP.80UP, SP.POP.1564, SP.POP.0014.IN, SP.DYN.AMRT,
SP.POP.TOTL.IN, SP.POP.65UP.IN))
```

Note: the Series Name column is included only to show the meaning of the Series Code values. This column can be dropped. At the end, the data should look as below.



Country Name	Country Code	SP.DYN.LE00.IN	SP.URB.TOTL	SP.POP.TOTL	SP.POP.80UP	SP.POP.1564.IN	SP.POP.0014.IN	SP.DYN.AMRT	SP.POP.TOTL.IN	SP.POP.65UP.IN
Afghanistan	AFG	63.37700	8535606	34413603	85552	18116800	15443807	455.4700	34413603	852996
Albania	ALB	78.02500	1654503	2880703	66965	1979175	537788	150.4100	2880703	363740
Algeria	DZA	76.09000	28146511	39728025	453741	25993589	11404930	191.6310	39728025	2329506
American Samoa	ASM	NA	48689	55812	NA	NA	NA	NA	NA	NA
Andorra	AND	NA	68919	78011	NA	NA	NA	NA	NA	NA
Angola	AGO	59.39800	17691524	27884381	69363	14113726	13136043	485.9310	27884381	634612

(d) Write code to show the top 5 countries with the lowest proportion of the population below 14 years old (i.e., SP.POP.0014.IN/SP.POP.TOTL) [Code, and list of 5 countries]

```
combo_demo %>% mutate(`fourteen and under` = SP.POP.0014.IN/SP.POP.TOTL.IN) %>%
select(`Country Name`, `fourteen and under`) %>% arrange(`fourteen and under`) %>%
slice(1:5)
`Country Name`        `fourteen and under`
  <chr>                              <dbl>
1 Hong Kong SAR, China               0.112
2 Macao SAR, China                   0.126
3 Singapore                          0.126
4 Japan                              0.130
5 Germany                            0.132
```