Data manipulation using dplyr

Here we look at the basic functionality offered by the dplyr[1] package for data manipulation.

The main commands in dplyr are the following:

- mutate adds new variables to a dataset.
- transmute creates a new dataset from transformations on an existing dataset. It is like mutate but "forgets" all the old variables.
- filter restricts the rows that we consider.
- select restricts the columns that we consider.
- group_by organizes the data into groups based on a variable.
- summarize aggregates multiple values (typically using a grouping variable).
- arrange allows the reordering of the dataset based on a variable.

Example 1: counties

To see some examples of using these, let's consider the following situation from the counties dataset:

1. We would like to find the percent of females on each state, and order the states based on that percent.
2. What we have at our disposal is the percent of females on each county, as well as the number of people on each county.
3. We can use this information to compute the number of females on each county, then add all those up by grouping per state.
4. We can also at the same time compute the total population on each state.
5. Finally, we can divide the two to find the percent of females on the state.
6. Then we can order by the state.

We will use the above commands in succession as follows:

1. We will use mutate to add a "count_female" variable that counts the number of females on each county.
2. We will group_by the state variable.
3. We will summarize to add up the female count and overall population count for each state. This will result in a dataset with *one row for each state*.
4. We will use mutate again to compute the percent of females on each state.
5. We will then use arrange to reorder the states.

Here is the overall code, though what you really want to do is look at each piece step by step (perhaps by piping it into the View() command):

```
counties %>%
    mutate(count_female = female * pop2010 / 100) %>%
    group_by(state) %>%
    summarize(
        counties = n(),
```

[1]https://dplyr.tidyverse.org/

```
        pop2010 = sum(pop2010),
        female = sum(count_female)
    ) %>%
    mutate(femalePercent = female/pop2010 * 100) %>%
    arrange(femalePercent)
```

We could store the result, or further process it by piping it into a graph or something of the kind. Let's suppose we had stored it in a variable femalesByState:

```
femalesByState <- counties %>% ......
femalesByState %>% gf_point(state~femalePercent)
```

Note that this graph does not actually order the states in increasing order, which is what we wanted. In order to achieve that, we must use the fct_reorder[2] command from the forcats package:

```
femalesByState %>% gf_point(fct_reorder(state, femalePercent)~femalePercent)
```

Example 2: smallest county per state

In this example we would like to collect one county from each state, namely the county with the smallest population in that state. In order to achieve that, we will perform the following steps:

1. Group the counties by state.
2. Arrange the counties by population size.
3. Summarize the counties, choosing the "first" for each group. At this point we have one county per state.
4. Include in that summary a column with the number of counties in the state.
5. Filter out states with only 1 county (DC).
6. Order by county population.
7. Create a new column containing the state name concatenated with the county name.
8. Draw a dotplot of the populations, with the new combined names.
9. Use fct_reorder to order the cases by population.
10. Set a logarithmic scale on the x axis.
11. Add the actual population counts to the data locations.

```
counties %>%
    group_by(state) %>%
    arrange(pop2010) %>%
    summarize(
        name = first(name),
        pop2010 = first(pop2010),
        counties = n()
    ) %>%
    filter(counties > 1)  %>%
    arrange(pop2010) %>%
    mutate(
        full = str_c(state, name, sep = " - ")
    ) %>%
```

[2] https://forcats.tidyverse.org/reference/fct_reorder.html

```
    gf_point(fct_reorder(full, pop2010)~pop2010) %>%
    gf_refine(scale_x_log10()) %>%
    gf_text(label=~pop2010, size=3, alpha=0.7, nudge_x=0.3, color="red")
```

Example 3: Smallest and largest county

We build on the previous example. We will now also compute the largest county per state. Some differences:

- We include both smallest and largest county points in the graph.
- We draw lines connecting the smallest and largest points.
- We also compute and place a dot at the median values on each state.
- We order the states by this median name.
- We join the medians with a line.

```
counties %>%
    group_by(state) %>%
    arrange(pop2010) %>%
    summarize(
        smallest_name = first(name),
        smallest = first(pop2010),
        largest_name = last(name),
        largest = last(pop2010),
        median = median(pop2010),
        counties = n()
    ) %>%
    filter(counties > 1) %>%
    mutate(
        state = fct_reorder(state, median)
    ) %>%
    gf_point(state~smallest) %>%
    gf_point(state~largest) %>%
    gf_linerangeh(state~smallest+largest) %>%
    gf_point(state~median, color="red") %>%
    gf_line(state~median, color="red", group=1) %>%
    gf_refine(scale_x_log10())
```