Data manipulation with dplyr

Daniel Stoxreiter

NB: The worksheet has beed developed and prepared by Lincoln Mullen. Source: Lincoln A. Mullen, *Computational Historical Thinking: With Applications in R (2018)*: http://dh-r.lincolnmullen.com.

The best way to learn R or computational history is to practice. These worksheets contain a series of questions designed to teach you about R or different computational methods. The worksheets are R Markdown documents that include text and code together. The places where you are expected to answer questions are marked like this.

```
(@) Can you make a plot from this dataset?
```

Beneath each question is a space to either create a code block or write an answer.

Aims of this worksheet

One of the key reasons to use R is to be able to manipulate data with ease. After completing this worksheet you will be able to work with the most commonly used data manipulation verbs provided by the dplyr and tidyr packages.

We will begin by loading the necessary packages and data. We will use the `methodists` dataset from the historydata package. This dataset contains membership figures for Methodist meetings (which were organized into districts, which were in turn organized into conferences) for the early nineteenth century.

```
library(tidyverse)
library(historydata)
load("C:/Users/Daniel/Documents/R_course/R-univie/lession5/methodists.rda")
```

Selecting columns (`select()`)

The first data manipulation verb that we are going to use is `select()`. This function lets us pass the names of the columns that we want to keep.

```
methodists %>%
  select(year, meeting, members_total)
```

Notice that we have not actually changed the data stored in `methodists` until we assign the changed data to a variable.

Read the documentation for this function, `?select`.

(1) Select the columns for `year`, `meeting`, as well as all columns that begin with the word `members_`.

```
methodists %>%
  select(year, meeting, contains("members_"))
```

(2) Remove the column `url`.

```r
methodists %>%
  select(-url)
```

Filtering rows (`filter()`)

The `select()` function lets us pick certain columns. The `filter()` function lets select certain rows based on logical conditions. For example, here we get the only the meetings where the total number of members is at greater than 1,000.

```r
methodists %>%
  filter(members_total > 1000)
```

(3) Get just the rows from New York in 1800.

```r
methodists %>%
  filter(state == "New York", year == 1800)
```

(4) Which Methodist meetings had only black members?

```r
methodists %>%
  filter(members_black > 0, members_white == 0)
```

Creating new columns (`mutate()`)

Very often one will want to create a new column based on other columns in the data. For instance, in our Methodist data, there is a column called `year`, but that column represents the year that the minutes were reported. The membership figures are actually for the previous year. Here we create a new column called `year_recorded`, where each value is one less than in `year`.

```r
methodists %>%
  mutate(year_recorded = year - 1) %>%
  select(year, year_recorded, meeting)
```

Notice that we chained the data manipulation functions using the pipe (`%>%`). This lets us create a pipeline were we can do many different manipulations in a row.

(5) Create two new columns, one with the percentage of white members, and one with the percentage of black members.

```r
methodists %>%
  mutate(percentage_white =  members_white / members_total * 100) %>%
  mutate(percentage_black = members_black / members_total * 100) %>%
  select(year, state, meeting, percentage_black, percentage_white)
```

Sorting columns (arrange())

Often we want to sort a data frame by one of its columns. This can be done with the verb `arrange()`. By default `arrange()` will sort from least to greatest; we can use the function `desc()` to sort from greatest to least. In this example, we sort the data frame to get the meetings with the highest number of white members.

```
methodists %>%
  arrange(desc(members_white))
```

(6) Which meetings had the highest number of black members? Select only the necessary columns so that the results print in a meaningful way.

```
methodists %>%
  arrange(desc(members_black)) %>%
  select(year, state, meeting, members_black)
```

(7) Which meetings had the high percentage of black members without being entirely black?

```
methodists %>%
  mutate(percentage_black = members_black / members_total * 100) %>%
  filter(percentage_black < 100) %>%
  arrange(desc(percentage_black)) %>%
  select(year, state, meeting, percentage_black, members_black, members_white)
```

Split-apply-combine (group_by())

Notice that in the example above the `arrange()` function sorted the entire data frame. So when we looked for the circuits with the largest number of members, we got rows from 1825, then 1830, then 1829, then 1830, and so on. What if we wanted to get the biggest circuit from each year?

We can solve this kind of problem with what Hadley Wickham calls the "split-apply-combine" pattern of data analysis. Think of it this way. First we can *split* the big data frame into separate data frames, one for each year. Then we can *apply* our logic to get the results we want; in this case, that means sorting the data frame. We might also want to get just the top one row with the biggest number of members. Then we can *combine* those split apart data frames into a new data frame.

Take a simple example using the `top_n()` function, which returns the top `n` (in this case, top 1) results for a particular column. After selecting a few columns, we get the row in the data frame which has the highest value for `members_black`.

```
methodists %>%
  select(year, meeting, members_total, members_black) %>%
  top_n(1, members_black)
```

We can change how that code works by using the `group_by()` function. Now we get the one row for each unique year in the dataset.

```
methodists %>%
  select(year, meeting, members_total, members_black) %>%
  group_by(year) %>%
  top_n(1, members_black)
```

We get the same results more concisely and reliably, though the steps of "split-apply-combine" are perhaps somewhat less easy to see.

(8) For each year, which was the biggest circuit?

```
methodists %>%
  select(year, meeting, members_total) %>%
  group_by(year) %>%
  top_n(1, members_total)
```

```
## # A tibble: 49 x 3
## # Groups:   year [49]
##     year meeting members_total
##    <int> <chr>           <int>
##  1  1786 Kent             1013
##  2  1787 Talbot           1601
##  3  1788 Sussex           1611
##  4  1789 Calvert          1852
##  5  1790 Calvert          1984
##  6  1791 Calvert          2089
##  7  1792 Calvert          1900
##  8  1793 Surry            1769
##  9  1794 Sussex           2354
## 10  1795 Calvert          1526
## # ... with 39 more rows
```

(9) For each year, which church had the biggest percentage of black members without being entirely black?

```
methodists %>%
  select(year, meeting, members_black, members_white, members_total) %>%
  mutate(percentage_black = members_black / members_total * 100) %>%
  filter(percentage_black < 100) %>%
  group_by(year) %>%
  top_n(1, percentage_black)
```

```
## # A tibble: 50 x 6
## # Groups:   year [49]
##     year meeting members_black members_white members_total percentage_black
##    <int> <chr>           <int>         <int>         <int>            <dbl>
##  1  1786 Calvert           316           295           611             51.7
##  2  1787 Charle~            53            34            87             60.9
##  3  1788 Calvert           842           505          1347             62.5
##  4  1789 Meckle~           692            98           790             87.6
##  5  1790 Annapo~           185           122           307             60.3
##  6  1791 Charle~           119            66           185             64.3
##  7  1792 George~           100            49           149             67.1
##  8  1793 Prince~           225            65           290             77.6
##  9  1793 George~           180            52           232             77.6
## 10  1794 Charle~           220            60           280             78.6
## # ... with 40 more rows
```

(10) For the year 1825, what was the biggest meeting in each conference? In each district?

```r
methodists %>%
  select(year, conference, district, meeting, members_total) %>%
  filter(year == 1825) %>%
  #group_by(conference) %>%
  group_by(district) %>%
  top_n(1, members_total)
```

```
## # A tibble: 74 x 5
## # Groups:   district [74]
##     year conference district    meeting        members_total
##    <int> <chr>      <chr>       <chr>                  <int>
##  1  1825 Ohio       Ohio        Youngstown               701
##  2  1825 Ohio       Portland    Mansfield                785
##  3  1825 Ohio       Lancaster   Muskingum               7775
##  4  1825 Ohio       Muskingum   Barnesville             1090
##  5  1825 Ohio       Scioto      Deer Creek              1022
##  6  1825 Ohio       Lebanon     Mad River               1419
##  7  1825 Ohio       Miami       Madison                  906
##  8  1825 Kentucky   Kenhawa     Little Kenhawa           648
##  9  1825 Kentucky   Augusta     Fleming                  930
## 10  1825 Kentucky   Green River Christian                734
## # ... with 64 more rows
```

(11) For each year, what was the biggest church in the Baltimore conference?

```r
methodists %>%
  select(year, conference, meeting, members_total) %>%
  filter(conference == "Baltimore") %>%
  group_by(year) %>%
  top_n(1, members_total)
```

```
## # A tibble: 33 x 4
## # Groups:   year [33]
##     year conference meeting members_total
##    <int> <chr>      <chr>           <int>
##  1  1802 Baltimore  Calvert          1612
##  2  1803 Baltimore  Calvert          2052
##  3  1804 Baltimore  Calvert          2457
##  4  1805 Baltimore  Calvert          2531
##  5  1806 Baltimore  Calvert          2421
##  6  1807 Baltimore  Calvert          2544
##  7  1808 Baltimore  Calvert          2273
##  8  1809 Baltimore  Calvert          2182
##  9  1810 Baltimore  Calvert          2303
## 10  1811 Baltimore  Calvert          2198
## # ... with 23 more rows
```

Summarizing or aggregating data (`summarize()`)

In the examples using `top_n()` we performed a very simple kind of data summary, where we took the single row with the biggest value in a given column. This essentially boiled many rows of a data frame down into a single row. We would like to be able to summarize or aggregate a data frame in other ways as well. For instance, we often want to take the sum or the mean of a given column. We can do this using the `summarize()` function in conjunction with the `group_by()` function.

In this example, we group by the year the minutes were taken. Then we find the total number of white members for each year.

```
methodists %>%
  group_by(year) %>%
  summarize(total_members_white = sum(members_white, na.rm = TRUE))
```

```
## # A tibble: 49 x 2
##     year total_members_white
##    <int>               <int>
##  1  1786               18291
##  2  1787               21949
##  3  1788               30557
##  4  1789               34425
##  5  1790               45983
##  6  1791               50580
##  7  1792               52079
##  8  1793               51486
##  9  1794               52794
## 10  1795               48121
## # ... with 39 more rows
```

Notice that we get one row in the recombined data frame for each group in the original data frame. The value in the new column is the result of a function (in this case, `sum()`) applied to the columns in each of the split apart data frames.

There is also a special case where we might want to know how many rows were in each of the split apart (or grouped) data frames. We can use the special `n()` function to get that count. (This is such a common thing to do that dplyr provides the special function `count()` to do this in an abbreviated way. You can look up that function's documentation to see how it works.)

```
methodists %>%
  group_by(year) %>%
  summarize(total_meetings = n())
```

```
## # A tibble: 49 x 2
##     year total_meetings
##    <int>          <int>
##  1  1786             51
##  2  1787             55
##  3  1788             76
##  4  1789             84
##  5  1790             99
##  6  1791            126
##  7  1792            135
##  8  1793            141
##  9  1794            146
## 10  1795            136
## # ... with 39 more rows
```

(12) How many meetings were there in each conference in each year since 1802?

```
methodists %>%
  filter(year >= 1802) %>%
  group_by(year, conference) %>%
  summarize(total_meetings = n())
```

```
## # A tibble: 389 x 3
## # Groups:   year [33]
##     year conference     total_meetings
##    <int> <chr>                   <int>
##  1  1802 Baltimore                  30
##  2  1802 New England                20
##  3  1802 New York                   36
##  4  1802 Philadelphia               43
##  5  1802 South Carolina             18
##  6  1802 Virginia                   30
##  7  1802 Western                    13
##  8  1803 Baltimore                  30
##  9  1803 New England                22
## 10  1803 New York                   39
## # ... with 379 more rows
```

(13) What is the average number of white, black, Indian and total members for each year since 1786?

```
methodists %>%
  filter(year >= 1786) %>%
  group_by(year) %>%
  summarise(avg_white = mean(members_white), avg_black = mean(members_black), avg_indian = mean(members_
```

```
## # A tibble: 49 x 5
##     year avg_white avg_black avg_indian avg_total
##    <int>     <dbl>     <dbl>      <dbl>     <dbl>
##  1  1786      359.      56.7          0      415.
##  2  1787      399.      70.6          0      470.
##  3  1788      402.     105.           0      507.
##  4  1789      410.     105.           0      515.
##  5  1790      464.     118            0      582.
##  6  1791      401.     104.           0      505.
##  7  1792      386.     103.           0      489.
##  8  1793      365.     102.           0      467.
##  9  1794      362.      95.2          0      457.
## 10  1795      354.      89.5          0      443.
## # ... with 39 more rows
```

Being able to create summaries like these is essential for visualizing the data.