week3

Hakan Mehmetcik

A grammar for data wrangling

In much the same way that ggplot2 presents a grammar for data graphics, the dplyr package presents a grammar for data wrangling.

Note

dplyr has five *verbs* for working with data in a data frame:

- `select()`: take a subset of the columns (i.e., features, variables)
- `filter()`: take a subset of the rows (i.e., observations)
- `mutate()`: add or modify existing columns
- `arrange()`: sort the rows
- `summarize()`: aggregate the data across rows (e.g., group it according to some criteria)

Overall, being able to combine these verbs with nouns (i.e., data frames) and adverbs (i.e., arguments) creates a flexible and powerful way to wrangle data!

select() and filter()

The two simplest of the five verbs are filter() and select(), which return a subset of the rows or columns of a data frame, respectively.

```
presidential |>
  select(name, party)
```

name	party
Eisenhower	Republican
Kennedy	Democratic
Johnson	Democratic
Nixon	Republican
Ford	Republican
Carter	Democratic
Reagan	Republican
Bush	Republican
Clinton	Democratic
Bush	Republican
Obama	Democratic
Trump	Republican

```r
presidential |>
  filter(party=="Republican")
```

name	start	end	party
Eisenhower	1953-01-20	1961-01-20	Republican
Nixon	1969-01-20	1974-08-09	Republican
Ford	1974-08-09	1977-01-20	Republican
Reagan	1981-01-20	1989-01-20	Republican
Bush	1989-01-20	1993-01-20	Republican
Bush	2001-01-20	2009-01-20	Republican
Trump	2017-01-20	2021-01-20	Republican

how about using both select and filter together

```r
presidential |>
  select(name, party) |>
  filter(party=="Republican")
```

name	party
Eisenhower	Republican
Nixon	Republican
Ford	Republican
Reagan	Republican
Bush	Republican

name	party
Bush	Republican
Trump	Republican

Combining the `filter()` and `select()` commands enables one to drill down to very specific pieces of information. For example, we can find which Democratic presidents served since *Watergate*.

```
presidential |>
  filter(year(start) > 1973 & party == "Democratic") |>
  select(name)
```

name
Carter
Clinton
Obama

`mutate()` and `rename()`

Frequently, in the process of conducting our analysis, we will create, re-define, and rename some of our variables. The functions `mutate()` and `rename()` provide these capabilities.

While we have the raw data on when each of these presidents took and relinquished office, we don't actually have a numeric variable giving the length of each president's term. Of course, we can derive this information from the dates given, and add the result as a new column to our data frame. This date arithmetic is made easier through the use of the lubridate package (now included as part of the tidyverse), which we use to compute the number of years (dyears()) that elapsed since during the interval() from the start until the end of each president's term.

In this situation, it is generally considered good style to create a new object rather than clobbering the one that comes from an external source. To preserve the existing presidential data frame, we save the result of mutate() as a new object called my_presidents.

```
my_presidents <- presidential |>
  mutate(term.length = interval(start, end) / dyears(1))
my_presidents
```

name	start	end	party	term.length
Eisenhower	1953-01-20	1961-01-20	Republican	8.000000

name	start	end	party	term.length
Kennedy	1961-01-20	1963-11-22	Democratic	2.836413
Johnson	1963-11-22	1969-01-20	Democratic	5.163587
Nixon	1969-01-20	1974-08-09	Republican	5.549624
Ford	1974-08-09	1977-01-20	Republican	2.450376
Carter	1977-01-20	1981-01-20	Democratic	4.000000
Reagan	1981-01-20	1989-01-20	Republican	8.000000
Bush	1989-01-20	1993-01-20	Republican	4.000000
Clinton	1993-01-20	2001-01-20	Democratic	8.000000
Bush	2001-01-20	2009-01-20	Republican	8.000000
Obama	2009-01-20	2017-01-20	Democratic	8.000000
Trump	2017-01-20	2021-01-20	Republican	4.000000

The mutate() function can also be used to modify the data in an existing column. Suppose that we wanted to add to our data frame a variable containing the year in which each president was elected. Our first (naïve) attempt might assume that every president was elected in the year before he took office. Note that mutate() returns a data frame, so if we want to modify our existing data frame, we need to overwrite it with the results.

```r
my_presidents <- my_presidents |>
  mutate(elected = year(start) - 1)
my_presidents
```

name	start	end	party	term.length	elected
Eisenhower	1953-01-20	1961-01-20	Republican	8.000000	1952
Kennedy	1961-01-20	1963-11-22	Democratic	2.836413	1960
Johnson	1963-11-22	1969-01-20	Democratic	5.163587	1962
Nixon	1969-01-20	1974-08-09	Republican	5.549624	1968
Ford	1974-08-09	1977-01-20	Republican	2.450376	1973
Carter	1977-01-20	1981-01-20	Democratic	4.000000	1976
Reagan	1981-01-20	1989-01-20	Republican	8.000000	1980
Bush	1989-01-20	1993-01-20	Republican	4.000000	1988
Clinton	1993-01-20	2001-01-20	Democratic	8.000000	1992
Bush	2001-01-20	2009-01-20	Republican	8.000000	2000
Obama	2009-01-20	2017-01-20	Democratic	8.000000	2008
Trump	2017-01-20	2021-01-20	Republican	4.000000	2016

Some entries in this data set are wrong, because presidential elections are only held every four years. Lyndon Johnson assumed the office after President John Kennedy was assassinated in

1963, and Gerald Ford took over after President Richard Nixon resigned in 1974. Thus, there were no presidential elections in 1962 or 1973, as suggested in our data frame. We should overwrite these values with NA's—which is how R denotes missing values. We can use the ifelse() function to do this. Here, if the value of elected is either 1962 or 1973, we overwrite that value with NA.1 Otherwise, we overwrite it with the same value that it currently has. In this case, instead of checking to see whether the value of elected equals 1962 or 1973, for brevity we can use the %in% operator to check to see whether the value of elected belongs to the vector consisting of 1962 and 1973.

```r
my_presidents <- my_presidents |>
  mutate(elected = ifelse(elected %in% c(1962, 1973), NA, elected))
my_presidents
```

name	start	end	party	term.length	elected
Eisenhower	1953-01-20	1961-01-20	Republican	8.000000	1952
Kennedy	1961-01-20	1963-11-22	Democratic	2.836413	1960
Johnson	1963-11-22	1969-01-20	Democratic	5.163587	NA
Nixon	1969-01-20	1974-08-09	Republican	5.549624	1968
Ford	1974-08-09	1977-01-20	Republican	2.450376	NA
Carter	1977-01-20	1981-01-20	Democratic	4.000000	1976
Reagan	1981-01-20	1989-01-20	Republican	8.000000	1980
Bush	1989-01-20	1993-01-20	Republican	4.000000	1988
Clinton	1993-01-20	2001-01-20	Democratic	8.000000	1992
Bush	2001-01-20	2009-01-20	Republican	8.000000	2000
Obama	2009-01-20	2017-01-20	Democratic	8.000000	2008
Trump	2017-01-20	2021-01-20	Republican	4.000000	2016

Finally, it is considered bad practice to use periods in the name of functions, data frames, and variables in R. Ill-advised periods could conflict with R's use of generic functions (i.e., R's mechanism for method overloading). Thus, we should change the name of the term.length column that we created earlier. We can achieve this using the rename() function. In this book, we will use snake_case for function and variable names.

Note

Don't use periods in the names of functions, data frames, or variables, as this can be confused with the object-oriented programming model.

```r
my_presidents <- my_presidents |>
  rename(term_length = term.length)
```

```r
my_presidents
```

name	start	end	party	term_length	elected
Eisenhower	1953-01-20	1961-01-20	Republican	8.000000	1952
Kennedy	1961-01-20	1963-11-22	Democratic	2.836413	1960
Johnson	1963-11-22	1969-01-20	Democratic	5.163587	NA
Nixon	1969-01-20	1974-08-09	Republican	5.549624	1968
Ford	1974-08-09	1977-01-20	Republican	2.450376	NA
Carter	1977-01-20	1981-01-20	Democratic	4.000000	1976
Reagan	1981-01-20	1989-01-20	Republican	8.000000	1980
Bush	1989-01-20	1993-01-20	Republican	4.000000	1988
Clinton	1993-01-20	2001-01-20	Democratic	8.000000	1992
Bush	2001-01-20	2009-01-20	Republican	8.000000	2000
Obama	2009-01-20	2017-01-20	Democratic	8.000000	2008
Trump	2017-01-20	2021-01-20	Republican	4.000000	2016

`arrange()`

The function `sort()` will sort a vector but not a data frame. The `arrange()` function sorts a data frame. In order to use `arrange()` on a data frame, you have to specify the data frame, and the column by which you want it to be sorted. You also have to specify the direction in which you want it to be sorted. Specifying multiple sort conditions will help break ties. To sort our `presidential` data frame by the length of each president's term, we specify that we want the column `term_length` in descending order.

```r
my_presidents |>
  arrange(desc(term_length))
```

name	start	end	party	term_length	elected
Eisenhower	1953-01-20	1961-01-20	Republican	8.000000	1952
Reagan	1981-01-20	1989-01-20	Republican	8.000000	1980
Clinton	1993-01-20	2001-01-20	Democratic	8.000000	1992
Bush	2001-01-20	2009-01-20	Republican	8.000000	2000
Obama	2009-01-20	2017-01-20	Democratic	8.000000	2008
Nixon	1969-01-20	1974-08-09	Republican	5.549624	1968
Johnson	1963-11-22	1969-01-20	Democratic	5.163587	NA
Carter	1977-01-20	1981-01-20	Democratic	4.000000	1976
Bush	1989-01-20	1993-01-20	Republican	4.000000	1988
Trump	2017-01-20	2021-01-20	Republican	4.000000	2016

name	start	end	party	term_length	elected
Kennedy	1961-01-20	1963-11-22	Democratic	2.836413	1960
Ford	1974-08-09	1977-01-20	Republican	2.450376	NA

A number of presidents completed either one or two full terms, and thus have the exact same term length (4 or 8 years, respectively). To break these ties, we can further sort by `party` and `elected`.

```
my_presidents |>
  arrange(desc(term_length), party, elected)
```

name	start	end	party	term_length	elected
Clinton	1993-01-20	2001-01-20	Democratic	8.000000	1992
Obama	2009-01-20	2017-01-20	Democratic	8.000000	2008
Eisenhower	1953-01-20	1961-01-20	Republican	8.000000	1952
Reagan	1981-01-20	1989-01-20	Republican	8.000000	1980
Bush	2001-01-20	2009-01-20	Republican	8.000000	2000
Nixon	1969-01-20	1974-08-09	Republican	5.549624	1968
Johnson	1963-11-22	1969-01-20	Democratic	5.163587	NA
Carter	1977-01-20	1981-01-20	Democratic	4.000000	1976
Bush	1989-01-20	1993-01-20	Republican	4.000000	1988
Trump	2017-01-20	2021-01-20	Republican	4.000000	2016
Kennedy	1961-01-20	1963-11-22	Democratic	2.836413	1960
Ford	1974-08-09	1977-01-20	Republican	2.450376	NA

`summarize()` with `group_by()`

Our last of the five verbs for single-table analysis is summarize(), which is nearly always used in conjunction with group_by(). The previous four verbs provided us with means to manipulate a data frame in powerful and flexible ways. But the extent of the analysis we can perform with these four verbs alone is limited. On the other hand, summarize() with group_by() enables us to make comparisons.

```
my_presidents |>
  summarize(
    N = n(),
    first_year = min(year(start)),
    last_year = max(year(end)),
    num_dems = sum(party == "Democratic"),
    years = sum(term_length),
    avg_term_length = mean(term_length)
  )
```

N	first_year	last_year	num_dems	years	avg_term_length
12	1953	2021	5	68	5.666667

The next two variables determine the first year that one of these presidents assumed office. This is the smallest year in the start column. Similarly, the most recent year is the largest year in the end column. The variable num_dems simply counts the number of rows in which the value of the party variable was Democratic. Finally, the last two variables compute the sum and average of the term_length variable. We see that 5 of the 12 presidents who served from 1953 to 2021 were Democrats, and the average term length over these 68 years was about 5.6 years.

```
my_presidents |>
  group_by(party) |>
  summarize(
    N = n(),
    first_year = min(year(start)),
    last_year = max(year(end)),
    num_dems = sum(party == "Democratic"),
    years = sum(term_length),
    avg_term_length = mean(term_length)
  )
```

party	N	first_year	last_year	num_dems	years	avg_term_length
Democratic	5	1961	2017	5	28	5.600000
Republican	7	1953	2021	0	40	5.714286