dplyr

Lecture 4

Federica Fusi

MSCA, UIC

Updated: 2021-02-09

DPLYR - Part 2

DPLYR

Dplyr is a package that allows you to interrogate your dataset in a more dynamic way. For instance, you might want to know how many crimes occur in a set of neighborhood in a certain time interval; you might want to know the average number of unemployment days by race and gender; the number of children in foster care by city and zip code, and so.

We used the dollar-sign syntax so far. But dplyr makes it easier to answer these questions and manipulate your data.

To use dplyr, there six verbs that you need to know:

- filter() to select cases based on their values
- arrange() to reorder the cases
- select() to select variables based on their names
- mutate() and transmute() to add new variables that are functions of existing variables
- summarise() to condense multiple values to a single value
- group_by() to group cases according to certain variables

Data

We are going to use some data from the fivethirtyeight package, which takes data from 538's articles and provide them in a tidy format for R users.

cabinet_turnover is a dataset reporting how long the members of the Cabinet stayed in charge as described in this article.

```
#install.packages("fivethirtyeight")

library("fivethirtyeight")

data = cabinet_turnover
```

summarise

summarise() to condense multiple values to a single value - in other words, it summarize across multiple rows.

summarise, Moderndive

summarise

Let's we want to find the average lenght that cabinet members stay in office.

```
data %>%
  summarize(average = mean(length, na.rm = T))
```

Or when it's the earliest that a cabinet member left its job

```
data %>%
  summarize(min_days = min(days, na.rm = T))
```

group_by

Summarize is the most useful when used with group_by as it provides the possibility to work with grouped summaries.

Let's see one example where we ask the article's main question: which presidents has the highest turnover? To answer, we need to know the average number of days that cabinet members were in charge, by president.

First, we need to group the observations by president and then calculate the average across all observations.

```
data %>%
  group_by(president) %>%
  summarise(presid_lenght = mean(length, na.rm = T))
```

Which president had the highest turnover?

group_by

What cabinet members is the most likely to resign after a short period in the administration? What cabinet member stay in charge, on average, the shortest amount of time?

```
data %>%
  group_by(position) %>%
  summarise(position_earliest = mean(days, na.rm = T)) %>%
  arrange(position_earliest) #<<

data %>%
  group_by(position) %>%
  summarise(position_length = mean(length, na.rm = T)) %>%
  arrange(position_length)

data %>%
  group_by(position) %>%
  summarise(position_length = mean(length, na.rm = T)) %>%
  arrange(desc(position_length))
```

summarize functions

You should be familiar with most of them:

- mean
- n
- median
- sd
- min
- max
- quantile(x, 0.25)
- sum

count

A couple of things that might be helpful to you moving forward.... Let's say you want to count how many cabinet members have left each president...

```
data %>%
  group_by(president) %>%
  summarise(count = n()) #<< #The function n is used to count the numbe
```

```
data %>%
  group_by(appointee) %>%
  summarise(count = n()) %>%
  arrange(desc(count)) # this would allow you to see if someone had more
```

What if you want to know if they had more than one appointment with the same president or another one?

Solutions

There are a few ways to do this and visualize your data

```
# This is the basic solution.
# Problem: we don't easily see if someone worked for more than one presi

data %>%
  group_by(appointee, president) %>%
  summarise(count = n()) %>%
  arrange(desc(count))

# Things might be easier if we arrange by appointee name as well
# The pring command allows you to print more than the standard 10 rows

print(
 data %>%
  group_by(appointee, president) %>%
  summarise(count = n()) %>%
  arrange(appointee, desc(count)), n= 300) # this would allow you to see
```

Solutions

```
# We could filter out the ones that had only one appointment
# We see all those who worked more than once for the same president
data %>%
  group_by(appointee, president) %>%
  summarise(count = n()) %>%
  filter(count >1) %>%
  arrange(appointee, desc(count))
```

Mutate

mutate() and transmute() to add new variables that are functions of existing variables - in other words, it allows you to perform operations on your columns.

Let's say that you want to know how long each cabinet member stayed in the administration as a percentage of the full adminstration term (4 years)

```
data2 =
  data %>%
  mutate(perc_length = (length / (365 * 4) * 100))
```

Note that you can always make longer pipe to see the data as you want them

```
data2 =
  data %>%
  mutate(perc_length = (length / (365 * 4) * 100)) %>%
  arrange(perc_length)
```

transmute

Transmute works like *mutate* but it creates a new column in a new dataset

```
data2 =
  data %>%
  transmute(perc_length = (length / (365 * 4) * 100)) %>% #<<
  arrange(perc_length)
```

mutate

As with the other functions, you can use 'mutate' within a pipe

How many cabinet members left before mid-administration?

```
data %>%
 mutate(perc_length = (length / (365 * 4) * 100)) %>%
 filter(perc_length <= 50)%>%
 summarise(count = n())
```

Group exercise

From R for Data Science This data frame contains all 336,776 flights that departed from New York City in 2013. The data comes from the US Bureau of Transportation Statistics, and is documented in ?flights.

```
library(nycflights13)

data = (flights)
```

What hour of day should you fly if you want to avoid arrival delays as much as possible? [intuitively, what do you expect to find?] (G1)

For each destination, compute the total minutes of delay. For each flight, compute the proportion of the total delay for its destination. (G2 and G3)

Find all destinations that are flown by at least two carriers. Use that information to rank the carriers. [hint: use the function n_distinct] (G4)

What hour of day should you fly if you want to avoid arrival delays as much as possible? [intuitively, what do you expect to find?]

```
# Question 1
data %>%
  group_by(hour) %>%
  summarise(arr_delay = mean(arr_delay, na.rm = T)) %>%
  arrange(arr_delay)
```

For each destination, compute the total minutes of delay. For each flight, compute the proportion of the total delay for its destination.

```
#Question 2

data1 =
data %>%
  filter(arr_delay >0) %>%
  group_by(dest) %>%
  mutate(tot_delay = sum(arr_delay), # by using mutate, we are reporting
         arr_delay_prop = arr_delay / tot_delay) # We are assuming that


data %>%
  filter(arr_delay > 0) %>%
  group_by(dest, origin, carrier, flight) %>%
  summarise(flight_delay = sum(arr_delay)) %>%
  group_by(dest) %>%
  mutate(arr_delay_prop = flight_delay / sum(flight_delay)) %>%
  arrange(dest, desc(arr_delay_prop)) %>%
  select(carrier, flight, origin, dest, arr_delay_prop)
```

Find all destinations that are flown by at least two carriers. Use that information to rank the carriers. [hint: use the function n_distinct]

```
#Question 3

data %>%
  group_by(dest) %>%
  mutate(n_carriers = n_distinct(carrier)) %>%
  filter(n_carriers >1) %>%
  group_by(carrier) %>%
  summarize(n_dest = n_distinct(dest)) %>%
  arrange(desc(n_dest))
```