Data Manipulation: Part 2

This tutorial will cover the following functions in the dplyr package:

* [mutate()](#mutate)
* [group\_by() with filter()](#groupfilter)
* [group\_by() with summarize()](#groupsum)

Advanced topic

* [Piping with %>%](#pipe)

## Make new columns with mutate()

In the first Data Manipulation tutorial we learned how to select just the columns we need by using the select() function from dplyr. But in some cases we may want to add a new column. For example, in the chicago\_air data set from the region5air package, we may want a column that is a one day lag value of the ozone column. To add a new column we use the mutate() function from the dplyr package. Again, the first parameter is the data frame. Then we use the lag() function to create a new column of lagged ozone values.

library(region5air)  
library(dplyr)  
data(chicago\_air)  
head(chicago\_air)

## date ozone temp solar month weekday  
## 1 2013-01-01 0.032 17 0.65 1 3  
## 2 2013-01-02 0.020 15 0.61 1 4  
## 3 2013-01-03 0.021 28 0.17 1 5  
## 4 2013-01-04 0.028 18 0.62 1 6  
## 5 2013-01-05 0.025 26 0.48 1 7  
## 6 2013-01-06 0.026 36 0.47 1 1

d <- mutate(chicago\_air, ozone\_lag1 = lag(ozone, n = 1))  
head(d)

## date ozone temp solar month weekday ozone\_lag1  
## 1 2013-01-01 0.032 17 0.65 1 3 NA  
## 2 2013-01-02 0.020 15 0.61 1 4 0.032  
## 3 2013-01-03 0.021 28 0.17 1 5 0.020  
## 4 2013-01-04 0.028 18 0.62 1 6 0.021  
## 5 2013-01-05 0.025 26 0.48 1 7 0.028  
## 6 2013-01-06 0.026 36 0.47 1 1 0.025

We may want to also calculate the difference between the ozone value and a standard value.

d <- mutate(d, ozone\_lag1 = lag(ozone, n = 1), diff\_standard = ozone - 0.075)  
head(d)

## date ozone temp solar month weekday ozone\_lag1 diff\_standard  
## 1 2013-01-01 0.032 17 0.65 1 3 NA -0.043  
## 2 2013-01-02 0.020 15 0.61 1 4 0.032 -0.055  
## 3 2013-01-03 0.021 28 0.17 1 5 0.020 -0.054  
## 4 2013-01-04 0.028 18 0.62 1 6 0.021 -0.047  
## 5 2013-01-05 0.025 26 0.48 1 7 0.028 -0.050  
## 6 2013-01-06 0.026 36 0.47 1 1 0.025 -0.049

## Filter by groups with group\_by() and filter()

So far we have selected columns with select(), added new columns with mutate(), and filtered an entire data frame with filter().

But sometimes when we filter we don't want to filter the entire data frame all at once. Suppose we are interested in the high values of ozone in the chicago\_air data frame. We can use the max() function to find the highest value.

filter(chicago\_air, ozone == max(ozone, na.rm = TRUE))

## date ozone temp solar month weekday  
## 1 2013-05-14 0.081 74 1.4 5 3

But that's for the entire year. Suppose you wanted to know the maximum value for each month. You could subset chicago\_air 12 times and make a new data frame for each month. Then find the maximum value for each data frame. But that would be very inefficient. What we really want to do is filter the data frame by first grouping all of the rows by month, then filtering on the highest ozone value in each month.

Below we create a data frame named ozone for illustration. It contains ozone values for one monitor, but for simplicity's sake it only has two days in June and two days in July.

ozone <- read.table(header=T, text='  
 day month value unit   
 1 June 0.040 ppm  
 2 June 0.050 ppm  
 1 July 0.045 ppm  
 2 July 0.052 ppm  
 ')

If we use the filter() function on ozone to get the maximum value, we'll get one row.

filter(ozone, value == max(value))

## day month value unit  
## 1 2 July 0.052 ppm

If we wanted to get the maximum value for each month, we would first use the group\_by() function.

ozone <- group\_by(ozone, month)

Now the ozone data frame is grouped by month. If we use the filter() function now, we'll get two rows because there are two months.

filter(ozone, value == max(value))

## Source: local data frame [2 x 4]  
## Groups: month [2]  
##   
## day month value unit  
## (int) (fctr) (dbl) (fctr)  
## 1 2 June 0.050 ppm  
## 2 2 July 0.052 ppm

What if we had two columns we wanted to group by? Here is another simple data frame to illustrate this situation.

ozone\_8h\_1h <- read.table(header=T, text='  
 day month duration value unit   
 1 June 1h 0.050 ppm  
 1 June 8h 0.040 ppm  
 2 June 1h 0.060 ppm  
 2 June 8h 0.050 ppm  
 1 July 1h 0.055 ppm  
 1 July 8h 0.045 ppm  
 2 July 1h 0.065 ppm  
 2 July 8h 0.052 ppm  
 ')

Now we might want to find the maximum value by month *and* by duration. We simply add duration in the group\_by() function.

ozone\_8h\_1h <- group\_by(ozone\_8h\_1h, month, duration)

Now the filter() function will return 4 rows, one for each month/duration pair.

filter(ozone\_8h\_1h, value == max(value))

## Source: local data frame [4 x 5]  
## Groups: month, duration [4]  
##   
## day month duration value unit  
## (int) (fctr) (fctr) (dbl) (fctr)  
## 1 2 June 1h 0.060 ppm  
## 2 2 June 8h 0.050 ppm  
## 3 2 July 1h 0.065 ppm  
## 4 2 July 8h 0.052 ppm

## Summarize by groups with group\_by() and summarize()

Filtering by grouped columns is appropriate for situations where we want to subset down to rows that meet a logical criteria. But sometimes we want to apply a summarizing function to grouped values.

Let's say we want to find the mean for each month and duration in the ozone\_8h\_1h data frame. We've use the group\_by() function already, ozone\_8h\_1h <- group\_by(ozone\_8h\_1h, month, value). We can't use filter() because there would be no value that satisfies the logical expression value == mean(value). Instead, we use the summarize() function.

summarize(ozone\_8h\_1h, mean = mean(value))

## Source: local data frame [4 x 3]  
## Groups: month [?]  
##   
## month duration mean  
## (fctr) (fctr) (dbl)  
## 1 July 1h 0.0600  
## 2 July 8h 0.0485  
## 3 June 1h 0.0550  
## 4 June 8h 0.0450

As you can see, the column "units" has been dropped. When you use summarize() you will only keep the columns that have been grouped-by and the column you create with the summarizing function. All other columns are dropped.

### When should you use filter() and when should you use summarise()?

* **Use filter() with group\_by()** when you want to keep only the rows that meet a logical criteria among groups. You will be using == because it's a logical filter. The output will retain all of the columns.
* **Use summarize() with group\_by()** when you need a summary of one column for each group. You will used = because you are creating a new column with a summarizing function. The output will only retain the columns you grouped-by and the one column you created with the summarizing function.

## Exercises

Exercises for this tutorial can be found here: http://rpubs.com/NateByers/DataManip2Exercises.

## Advanced Topic

### Piping with %>%

dplyr takes advantage of the pipe operator, %>%, from the magrittr package.

The %>% is a symbol you can use to chain dplyr functions together. This is convenient because it eliminates the necessity of creating new data frames, or re-assigning values to the same data frame.

Here's an example of a data manipulation process.

data(airdata)  
  
# filter down to ozone  
ozone <- filter(airdata, parameter == "44201")  
  
# select only certain columns  
ozone <- select(ozone, site, datetime, duration, value, unit, poc)  
  
# group by certain columns  
ozone <- group\_by(ozone, site, duration, poc)  
  
# find mean value for each group  
ozone <- summarize(ozone, mean = mean(value, na.rm = FALSE))  
  
head(ozone)

## Source: local data frame [6 x 4]  
## Groups: site, duration [6]  
##   
## site duration poc mean  
## (chr) (int) (int) (dbl)  
## 1 840170310001 60 1 0.02534631  
## 2 840170310032 60 1 0.03259597  
## 3 840170310064 60 1 0.02392692  
## 4 840170310076 60 1 0.02375103  
## 5 840170311003 60 2 0.02771658  
## 6 840170311601 60 1 0.02588920

With %>% you can start with the data frame and then write each dplyr function without supplying the data frame.

ozone <- airdata %>%  
 filter(parameter == "44201") %>%  
 select(site, datetime, duration, value, unit, poc) %>%  
 group\_by(site, duration, poc) %>%  
 summarize(mean = mean(value, na.rm = FALSE))  
  
head(ozone)

## Source: local data frame [6 x 4]  
## Groups: site, duration [6]  
##   
## site duration poc mean  
## (chr) (int) (int) (dbl)  
## 1 840170310001 60 1 0.02534631  
## 2 840170310032 60 1 0.03259597  
## 3 840170310064 60 1 0.02392692  
## 4 840170310076 60 1 0.02375103  
## 5 840170311003 60 2 0.02771658  
## 6 840170311601 60 1 0.02588920