Summarizing with dplyr

## Learning Objectives

* Create new columns using mutate
* Summarize data using summarize
* Count number of observations using n()
* Group data by variable(s) with group\_by

At this point we have only used dplyr to subset and organize our data. But of course we can also create new data. And the true power of dplyr is revealed when we perform these operations by groups.

### Create new columns with mutate

To create a new column in the data frame, we use mutate. Let's create a new column that is the number of weeks since the article was published.

research <- mutate(research,  
 weeksSincePublished = daysSincePublished / 7)

We can instantly reference the new variables we have created. For example, we can create a variable yearsSincePublished referencing the newly created weeksSincePublished.

research <- mutate(research,  
 weeksSincePublished = daysSincePublished / 7,  
 yearsSincePublished = weeksSincePublished / 52)  
select(research, contains("Since")) %>% slice(1:10)

daysSincePublished weeksSincePublished yearsSincePublished  
1 2628 375.4286 7.219780  
2 2593 370.4286 7.123626  
3 2684 383.4286 7.373626  
4 2684 383.4286 7.373626  
5 2628 375.4286 7.219780  
6 2628 375.4286 7.219780  
7 2656 379.4286 7.296703  
8 2656 379.4286 7.296703  
9 2628 375.4286 7.219780  
10 2628 375.4286 7.219780

### Summarize data using summarize

We use mutate when the result has the same number of rows as the original data. When we need to reduce the data to a single summary statistic, we can use summarize. For example, let's calculate a summary statistic which is the mean number of PLOS comments.

summarize(research, plos\_mean = mean(plosCommentCount))

plos\_mean  
1 0.2642681

And we can additional statistics, like the standard deviation:

summarize(research, plos\_mean = mean(plosCommentCount),  
 plos\_sd = sd(plosCommentCount))

plos\_mean plos\_sd  
1 0.2642681 1.230676

Notice that this creates a second column in the data frame result.

And of course we can pipe input to summarize. Let's calculate these statistics specifically for the articles in PLOS One published in 2007.

research %>% filter(journal == "pone", year == 2007) %>%  
 summarize(plos\_mean = mean(plosCommentCount),  
 plos\_sd = sd(plosCommentCount))

plos\_mean plos\_sd  
1 0.8315704 2.033141

Lastly, since it is often useful to know how many observations, in this case articles, are present in a given subset, dplyr provides the convenience function n().

research %>% filter(journal == "pone", year == 2007) %>%  
 summarize(plos\_mean = mean(plosCommentCount),  
 plos\_sd = sd(plosCommentCount),  
 num = n())

plos\_mean plos\_sd num  
1 0.8315704 2.033141 1229

### Summarizing per group with group\_by

The function summarize is most powerful when applied to groupings of the data. dplyr makes the code much easier to write, understand, and extend.

Recall the function we wrote earlier to calculate the mean of a metric for each level of a factor.

mean\_metric\_per\_var <- function(metric, variable) {  
 result <- numeric(length = length(levels(variable)))  
 names(result) <- levels(variable)  
 for (v in levels(variable)) {  
 result[v] <- mean(metric[variable == v])  
 }  
 return(result)  
}

Which we ran as the following.

mean\_metric\_per\_var(research$backtweetsCount, research$journal)

pbio pcbi pgen pmed pntd pone   
0.05811321 0.12657291 0.06547251 0.31104199 0.02576490 0.49303878   
 ppat   
0.02604524

We can perform the same operation by combining summarize with group\_by

research %>% group\_by(journal) %>%  
 summarize(tweets\_mean = mean(backtweetsCount))

Source: local data frame [7 x 2]  
  
 journal tweets\_mean  
1 pbio 0.05811321  
2 pcbi 0.12657291  
3 pgen 0.06547251  
4 pmed 0.31104199  
5 pntd 0.02576490  
6 pone 0.49303878  
7 ppat 0.02604524

Conveniently it returns the result as a data frame. And if we want to further group it by another factor, we can just add it to the group\_by function.

research %>% group\_by(journal, year) %>%  
 summarize(tweets\_mean = mean(backtweetsCount))

Source: local data frame [42 x 3]  
Groups: journal  
  
 journal year tweets\_mean  
1 pbio 2003 0.000000000  
2 pbio 2004 0.000000000  
3 pbio 2005 0.011363636  
4 pbio 2006 0.010869565  
5 pbio 2007 0.004926108  
6 pbio 2008 0.030456853  
7 pbio 2009 0.005524862  
8 pbio 2010 0.367231638  
9 pcbi 2005 0.000000000  
10 pcbi 2006 0.000000000  
.. ... ... ...

In the function we wrote to do this manually, we would have had to write another for loop!

### Challenges

## Summarizing the number of tweets per journal

Create a new data frame, tweets\_per\_journal, that for each journal contains the total number of articles, the mean number of tweets received by articles in that journal, and the standard error of the mean (SEM) of the number of tweets. The SEM is the standard deviation divided by the square root of the sample size (i.e. the number of articles).

tweets\_per\_journal <- research %>%  
 group\_by(journal) %>%  
 summarize(num = n(),  
 mean = mean(backtweetsCount),  
 sem = sd(backtweetsCount) / sqrt(num))  
tweets\_per\_journal

Source: local data frame [7 x 4]  
  
 journal num mean sem  
1 pbio 1325 0.05811321 0.020153395  
2 pcbi 1351 0.12657291 0.052177184  
3 pgen 1619 0.06547251 0.020408525  
4 pmed 643 0.31104199 0.187868371  
5 pntd 621 0.02576490 0.009057697  
6 pone 14078 0.49303878 0.034484187  
7 ppat 1459 0.02604524 0.008807428