STAT 209: Lab5

Code

Five Data-Wrangling Verbs

Goal

Gain practice with the five “fundamental verbs” that are the building blocks in the “grammar of data wrangling”, as implemented in the `dplyr` package.

The verbs are:

1. `filter()`
2. `select()`
3. `mutate()`
4. `arrange()`
5. `summarize()`

Resources

You will probably want to look at the reference sheet (http://colindawson.net/stat209/resources/dplyr-quick-reference.pdf) or the slides (http://colindawson.net/stat209/slides/11-grammar-of-wrangling.pdf) from time to time. Remember that knowing how to look things up is an important skill! Nobody memorizes everything.

The Data

We'll look some more at the `babynames` dataset for this lab. Let's make sure it (as well as the `tidyverse` package) is loaded in our Markdown document.

Code:

Code

```
## Observations: 1,924,665
## Variables: 5
## $ year <dbl> 1880, 1880, 1880, 1880, 1880, 1880, 1880, 1880, 1880, 1880,…
## $ sex  <chr> "F", "F", "F", "F", "F", "F", "F", "F", "F", "F", "F", "F",…
## $ name <chr> "Mary", "Anna", "Emma", "Elizabeth", "Minnie", "Margaret", …
## $ n    <int> 7065, 2604, 2003, 1939, 1746, 1578, 1472, 1414, 1320, 1288,…
## $ prop <dbl> 0.07238359, 0.02667896, 0.02052149, 0.01986579, 0.01788843,…
```

Extracting subsets with `filter()`

Recall that `filter()` allows us to extract a subset of cases from a dataset, by checking for a particular criterion. Let's extract female "Bella"s (you might be able to do this without peeking at the code below at this point) and display the first few rows with `head()`.

Code:

Code

```
## # A tibble: 6 x 5
##    year sex   name      n     prop
##   <dbl> <chr> <chr> <int>    <dbl>
## 1  1880 F     Bella    13 0.000133
## 2  1881 F     Bella    24 0.000243
## 3  1882 F     Bella    16 0.000138
## 4  1883 F     Bella    17 0.000142
## 5  1884 F     Bella    31 0.000225
## 6  1885 F     Bella    25 0.000176
```

Selecting variables with `select()`

Recall that `select()` allows us to extract certain columns from a dataset, either by listing each variable name we want to include as a separate argument, or by listing each variable name we want to exclude, or by defining a condition for inclusion/exclusion.

Let's start with the full `babynames` dataset (so, not just Bellas) and display the first few rows, retaining only the `year`, `name` and `n` variables.

Code:

Code

```
## # A tibble: 6 x 3
##    year name          n
##   <dbl> <chr>     <int>
## 1  1880 Mary       7065
## 2  1880 Anna       2604
## 3  1880 Emma       2003
## 4  1880 Elizabeth  1939
## 5  1880 Minnie     1746
## 6  1880 Margaret   1578
```

Now create a new dataset called `Bellas` that retains just `year` and `n` for the first few years of female Bellas, by chaining `filter()` and `select()` together with pipes, and assigning the result. Check that the result looks as it should using `head()`.

Code:

Code

```
## # A tibble: 6 x 2
##    year     n
##   <dbl> <int>
## 1  1880    13
## 2  1881    24
## 3  1882    16
## 4  1883    17
## 5  1884    31
## 6  1885    25
```

Defining new variables with `mutate()`

Suppose we want to split the set of name/sex pairs into those that were “popular” in a given year, defined as being a name that was assigned to at least 1% of all babies of a particular sex (as assigned at birth) that year, and those that were not so popular. We can define a new binary variable based on the `prop` variable using `mutate()`.

Code:

Code

```
## # A tibble: 6 x 6
##    year sex   name          n   prop popular
##   <dbl> <chr> <chr>     <int>  <dbl> <lgl>
## 1  1880 F     Mary       7065 0.0724 TRUE
## 2  1880 F     Anna       2604 0.0267 TRUE
## 3  1880 F     Emma       2003 0.0205 TRUE
## 4  1880 F     Elizabeth  1939 0.0199 TRUE
## 5  1880 F     Minnie     1746 0.0179 TRUE
## 6  1880 F     Margaret   1578 0.0162 TRUE
```

If we decide the name a variable is given, we can replace it using the `rename()` function. For example, let's rename `popular` to `is_popular`. This function has the following syntax:

```
rename(dataset, newname1 = oldname1, newname2 = oldname2, ...)
```

Code

Code

```
## # A tibble: 6 x 6
##    year sex   name          n   prop is_popular
##   <dbl> <chr> <chr>     <int>  <dbl> <lgl>
## 1  1880 F     Mary       7065 0.0724 TRUE
## 2  1880 F     Anna       2604 0.0267 TRUE
## 3  1880 F     Emma       2003 0.0205 TRUE
## 4  1880 F     Elizabeth  1939 0.0199 TRUE
## 5  1880 F     Minnie     1746 0.0179 TRUE
## 6  1880 F     Margaret   1578 0.0162 TRUE
```

Exercise 1 Create a dataset called `PopularBabynames` that includes only those names that were "popular" in the given year. Use the new `is_popular` variable to do the filtering, and then remove the variable from the filtered dataset since it is now a "constant".

Solution:

Code

```
## # A tibble: 6 x 5
##    year sex   name          n   prop
##   <dbl> <chr> <chr>     <int>  <dbl>
## 1  1880 F     Mary       7065 0.0724
## 2  1880 F     Anna       2604 0.0267
## 3  1880 F     Emma       2003 0.0205
## 4  1880 F     Elizabeth  1939 0.0199
## 5  1880 F     Minnie     1746 0.0179
## 6  1880 F     Margaret   1578 0.0162
```

Sorting data with `arrange()`

We can easily see at what point the largest share of births went to a single name by sorting the dataset by `prop`. We can use `arrange()` for this. To arrange in descending order so that the *most* popular name is at the top, use the `desc()` helper function around the variable name.

Code:

Code

```
## # A tibble: 6 x 5
##    year sex   name        n   prop
##   <dbl> <chr> <chr>   <int>  <dbl>
## 1  1880 M     John     9655 0.0815
## 2  1881 M     John     8769 0.0810
## 3  1880 M     William  9532 0.0805
## 4  1883 M     John     8894 0.0791
## 5  1881 M     William  8524 0.0787
## 6  1882 M     John     9557 0.0783
```

Exercise 2 Describe precisely what the `prop` variable is telling us here. What does it mean for a name to be "first" in this list?

Solution:

Code

Exercise 3 Find the most popular male and female name in your birth year.

Possible Solution:

Code

```
## # A tibble: 7,364 x 5
##     year sex   name            n   prop
##    <dbl> <chr> <chr>       <int>  <dbl>
##  1  1982 M     Michael     68228 0.0362
##  2  1982 M     Christopher 59225 0.0314
##  3  1982 M     Matthew     46060 0.0244
##  4  1982 M     Jason       40624 0.0215
##  5  1982 M     David       40451 0.0214
##  6  1982 M     James       38872 0.0206
##  7  1982 M     Joshua      38027 0.0202
##  8  1982 M     John        34699 0.0184
##  9  1982 M     Robert      34421 0.0182
## 10  1982 M     Daniel      32653 0.0173
## # … with 7,354 more rows
```

Code

Exercise 4 Choose a name, sex pair you like (perhaps your name and sex you identify with, if you identify with one in this dataset). Find the birth year when that name was most popular for babies assigned that sex.

Possible Solution:

Code

```
## # A tibble: 127 x 5
##     year sex   name      n    prop
##    <dbl> <chr> <chr> <int>   <dbl>
##  1  2004 M     Colin  5122 0.00242
##  2  2003 M     Colin  4876 0.00232
##  3  2005 M     Colin  4531 0.00213
##  4  2006 M     Colin  3858 0.00176
##  5  2009 M     Colin  3655 0.00172
##  6  2008 M     Colin  3728 0.00171
##  7  2010 M     Colin  3486 0.00170
##  8  2007 M     Colin  3608 0.00163
##  9  2011 M     Colin  3265 0.00161
## 10  2002 M     Colin  3315 0.00160
## # … with 117 more rows
```

Code

Exercise 5 Find the birth year in which the greatest *number* of babies were born with your name and sex (or name and sex of choice).
Explain why it *could* be different than the year in the last question.

Possible Solution:

Code

```
## # A tibble: 127 x 5
##     year sex   name      n    prop
##    <dbl> <chr> <chr> <int>   <dbl>
##  1  2004 M     Colin  5122 0.00242
##  2  2003 M     Colin  4876 0.00232
##  3  2005 M     Colin  4531 0.00213
##  4  2006 M     Colin  3858 0.00176
##  5  2008 M     Colin  3728 0.00171
##  6  2009 M     Colin  3655 0.00172
##  7  2007 M     Colin  3608 0.00163
##  8  2010 M     Colin  3486 0.00170
##  9  2002 M     Colin  3315 0.00160
## 10  2011 M     Colin  3265 0.00161
## # … with 117 more rows
```

Code

Exercise 6 Think about how you might address the following question, without actually trying to do it now: In which year was a particular name (pick any name) the most balanced between males and females; that is, when was the number of male and female births for that name closest to a 50/50 split?

Possible Solution:

Code

Calculating summary statistics with `summarize()`

The `summarize()` verb works a little bit differently than the other four verbs. Whereas `filter()`, `select()`, `mutate()`, and `arrange()` take in a dataset where the rows are cases and the columns are variables and return a dataset in the same form, `summarize()` takes a dataset where the rows are cases and the columns are variables and returns a dataset with just one row (at least, when it is used by itself), where the columns are summary statistics (things like means, standard deviations, etc.) calculated from all the cases in the input.

Tip: When using `summarize()`, it is almost always desirable to return as one of the summary statistics the number of cases in the set being summarized. Among other things, this can be a quick way to alert you to errors. The `n()` function (called with no arguments) is a special helper function that does this.

NB: The `babynames` data contains a *variable* called `n`. Don't confuse this variable with the function `n()`. In fact, to prevent confusion, rename the `n` variable to `num_births`

Code

Code

```
## # A tibble: 6 x 6
##    year sex   name      num_births   prop is_popular
##   <dbl> <chr> <chr>          <int>  <dbl> <lgl>
## 1  1880 F     Mary            7065 0.0724 TRUE
## 2  1880 F     Anna            2604 0.0267 TRUE
## 3  1880 F     Emma            2003 0.0205 TRUE
## 4  1880 F     Elizabeth       1939 0.0199 TRUE
## 5  1880 F     Minnie          1746 0.0179 TRUE
## 6  1880 F     Margaret        1578 0.0162 TRUE
```

Suppose we want to find the year in which the name "Bella" hit its peak for females. We can take the `Bellas` dataset we created above, and use `summarize()` together with the `max()` function to answer this question.

Code:

Code

```
## # A tibble: 1 x 2
##   num_rows most_bellas
##      <int>       <int>
## 1      138        5121
```

Hmm… This tells us how many female Bellas were born in that name's peak year for females, but it doesn't actually tell us the year when that happened. We can use the `which.max()` function to get the relevant row number, and pull out the `year` in that position using square bracket element selection syntax, as follows.

Code

```
## # A tibble: 1 x 3
##   num_rows peak_number peak_year
##      <int>       <int>     <dbl>
## 1      138        5121      2010
```

Exercise 7 Explain what the `num_rows` value tells us in context.

Possible Solution:

Code

Exercise 8 Use `summarize()` instead of `arrange()` to redo exercises 3 and 4.

Possible Solution:

Code

```
## # A tibble: 1 x 3
##   num_rows peak_prop most_popular_name
##      <int>     <dbl> <chr>
## 1     7364    0.0362 Michael
```

Code

```
## # A tibble: 1 x 3
##   num_rows peak_prop peak_year
##      <int>     <dbl>     <dbl>
## 1      127   0.00242      2004
```

Piping Practice

And now for the cake-decorating portion of the lab. Just kidding.

Recall that when we write

```
dataset %>% verb(arguments)
```

this is equivalent to writing

```
verb(dataset, arguments)
```

More generally,

```
some_function(main_argument, other_arguments)
```

is rewritten as

```
main_argument %>% some_function(other_arguments)
```

With just one function it's not clear that the pipe syntax is any clearer, but when we start chaining operations together, writing the verbs from left to right instead of from inside out (which is how we'd have to do it without the pipe) makes the code a whole lot easier to read.

Exercise 9 Re-write the following mess of a command, which displays a list of the top ten years in which male Colins were born in descending order of the number of male babies named Colin that year, using the pipe operator.

Code

```
## # A tibble: 10 x 2
##     year num_births
##    <dbl>      <int>
##  1  2004       5122
##  2  2003       4876
##  3  2005       4531
##  4  2006       3858
##  5  2008       3728
##  6  2009       3655
##  7  2007       3608
##  8  2010       3486
##  9  2002       3315
## 10  2011       3265
```

Possible solution:

Code

Getting Credit

Send a DM to me and Chris Ikeokwu in Slack with the following: * Your code to find the peak birth year for your (chosen) name in two ways: one using `arrange()` and one using `summarize()`. * What aspects of the data-wrangling we've done so far do you feel most comfortable with after doing this lab? * What aspects do you feel least comfortable with? * Write the honor pledge, signifying that you completed the whole lab (and not just the parts where you had to turn in something)