Data 8
Fall 2016

Discussion

Categories and Grouping

Data 8 has opened up a marble store where we sell small bags of marbles in groups of different sizes and colors. Our table, called *marbles*, is as follows:

Color	Amount	Price
Red	4	1.30
Green	6	1.20
Blue	12	2.0
Red	7	1.75
Green	9	1.40
Green	2	1.0

The *marbles* table currently has three columns: the color, the amount in each bag, and the price of each bundle of marbles available for purchase.

Group:

- Takes in one or more arguments. The first argument should be a column name for the column we would like to group the values of. **Grouping over a column will produce one row for each unique value in the column.**
- By default, if no second argument is included, *count* will be the function used to group the elements in the column.
- Other functions that can be used to group over are *sum*, *mean*, etc.

Question 1: Notice that our table contains multiple rows containing information about marbles of the same color. We would like to figure out how many marbles of each color we have for sale in total, and what the cost would be to purchase all marbles of each unique color.

1.1 First, write a line of code that will return a new table which displays the total number of marbles of each color.

1.2 Now let's say we want to find the biggest, most expensive bundle of marbles of each color. Write a line of code that will return a new table with this information in it.

1.3 Last but not least, write a line of code which will return a new table with the combined total numbers of marbles and costs for each unique color.

****

We have updated our inventory of marbles, and have begun selling marbles of different shapes as well as colors. A table representing our new inventory is as follows:

Color	Shape	Amount	Price
Red	Round	4	1.30
Green	Rectangular	6	1.20
Blue	Rectangular	12	2.00
Red	Round	7	1.75
Green	Rectangular	9	1.40
Green	Round	2	1.00

Groups

So, we have seen group, and now it's time to talk about Groups. As you might notice, the *groups* function's name bears an awful resemblance to the *group* function which we have already talked about, and this is for a good reason. The first argument of *groups* should take in a list of columns you would like to group over. Now, instead of each unique element from a category being in its own row, now it's the *unique combinations of elements* from the categories we provide that must have their own rows.

Question 2: Fill in the following table that will result from calling $marbles.groups(['Color','Shape'], sum)$.

Color	Shape	Amount _____	Price _____
Red	Round	_____	_____
Green	Rectangular	_____	_____
Blue	Rectangular	_____	_____
Green	Round	_____	_____

Pivot
At first, *pivot* might seem like the more confusing of the two, but its similarity to *groups* is very interesting.

- The *pivot* function takes in three arguments, the category which will be unique on the columns, the category which will be unique on the rows, and the values on which we want to aggregate.
- An optional fourth argument is used to specify the aggregation function we would like to use. Without this argument, the default will be $'count'$, as for *group* and *groups*.

We will continue with the same table as before, copied below for your convenience.

Color	Shape	Amount	Price
Red	Round	4	1.30
Green	Rectangular	6	1.20
Blue	Rectangular	12	2.00
Red	Round	7	1.75
Green	Rectangular	9	1.40
Green	Round	2	1.00

Question 3: Create a pivot table on the types of marbles, finding the average price for each type. Assume the *avg* function is given to you and finds the average over a list of inputs.

Color		

****

Assume that there was a sale, and we get different amounts of discount based on how many marbles we are trying to buy from this store. The *sales* table is as follows:

Count	Discount
2	5%
4	10%
6	20%
7	30%
9	35%
12	40%

Now, our goal is to get our computer to match up how much discount is applied to each bunch. Join will do just the trick for us.

Join
Join takes in three arguments: the column on which to join from the first table, the second table to join over (can be the same as the first), and the name of the column in the second table on which to join.

Question 4: Write a line of code that will compute the table for the amount of discount applied to each bundle of marbles. The resultant table from your code should look like this:

Amount	Color	Shape	Price	Discount
4	Red	Round	1.30	10%
6	Green	Rectangular	1.20	2%
12	Blue	Rectangular	2.0	40%
7	Red	Round	1.75	30%
9	Green	Rectangular	1.40	35%
2	Green	Round	1.0	5%