Eddie Aguilar

Add new variables with mutate()

A data set often contains information that you can use to compute new variables. mutate() helps you compute those variables. Since mutate() always adds new columns to the end of a dataset, we'll start by creating a narrow dataset which will let us see the new variables (If we added new variables to flights, the new columns would run off the side of your screen, which would make them hard to see).

select()

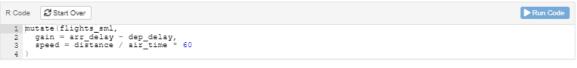
You can select a subset of variables by name with the select() function in dplyr. Run the code below to see the narrow data set that select() creates.



arr_delay <dbl></dbl>	dep_delay <dbl></dbl>	distance <dbl></dbl>	air_time <dbl></dbl>
11	2	1400	227
20	4	1416	227
33	2	1089	160
-18	-1	1576	183
-25	-6	762	116
12	-4	719	150
19	-5	1065	158
-14	-3	229	53
-8	-3	944	140
8	-2	733	138
1-10 of 1,000 rows		Previous 1 2 3	4 5 6 100 Next

~ mutate()

The code below creates two new variables with dplyr's mutate() function. mutate() returns a new data frame that contains the new variables appended to a copy of the original data set. Take a moment to imagine what this will look like, and then click "Run Code" to find out.



arr_delay <dbl></dbl>	_ , , , ,		air_time <dbl></dbl>	gain <dbl></dbl>	speed <dbl></dbl>
11	2	1400	227	9	370.0441

Note that when you use mutate() you can create multiple variables at once, and you can even refer to variables that are created earlier in the call to create other variables later in the call:

```
mutate(flights_sml,
  gain = arr_delay - dep_delay,
  hours = air_time / 60,
  gain_per_hour = gain / hours
)
```

arr_delay <dbl></dbl>	dep_delay <dbl></dbl>	distance <dbl></dbl>	air_time <dbl></dbl>	gain <dbl></dbl>	hours <dbl></dbl>
11	2	1400	227	9	3.7833333
20	4	1416	227	16	3.7833333
33	2	1089	160	31	2.6666667
-18	-1	1576	183	-17	3.0500000
-25	-6	762	116	-19	1.9333333
12	-4	719	150	16	2.5000000
19	-5	1065	158	24	2.6333333
-14	-3	229	53	-11	0.8833333
-8	-3	944	140	-5	2.3333333
8	-2	733	138	10	2.3000000
1-10 of 1,000 rows 1-6 of 7 columns			Previous 1	2 3	4 5 6 100 Next

transmute()

mutate() will always return the new variables appended to a copy of the original data. If you want to return only the new variables, use transmute(). In the code below, replace mutate() with transmute() and then spot the difference in the results.

R Code Start Over Solution	▶ Run Code ☑ Submit Answer
1 transmute(flights, 2 gain = arr delay - dep_delay, 3 hours = air time / 60,	
4 gain_per_hour = gain / hours 5)	

gain_per_hour <dbl></dbl>	hours <dbl></dbl>	gain <dbl></dbl>
2.3788546	3.7833333	9
4.2290749	3.7833333	16
11.6250000	2.6666667	31
-5.5737705	3.0500000	-17
-9.8275862	1.9333333	-19
6.4000000	2.5000000	16
9.1139241	2.6333333	24
-12.4528302	0.8833333	-11

Useful mutate functions

You can use any function inside of mutate() so long as the function is vectorised. A vectorised function takes a vector of values as input and returns a vector with the same number of values as output.

Over time, I've found that several families of vectorised functions are particularly useful with [mutate()]:

- Arithmetic operators: +, -, *, /, ^. These are all vectorised, using the so called "recycling rules". If one parameter is shorter than the other, it will automatically be repeated multiple times to create a vector of the same length. This is most useful when one of the arguments is a single number: air_time / 60, hours * 60 + minute, etc.
- Modular arithmetic: \$/\$ (integer division) and \$\$ (remainder), where x = y + (x */\$ y) + (x *§ y). Modular arithmetic is a handy tool because it allows you to break integers up into pieces. For example, in the flights dataset, you can compute hour and minute from dep_time with:

```
transmute(flights,
  dep_time,
  hour = dep_time %/% 100,
  minute = dep_time %% 100
)
```

dep_time <int></int>	hour <dbl></dbl>	minute <dbl></dbl>
517	5	17
533	5	33
542	5	42
544	5	44
554	5	54
554	5	54
555	5	55
557	5	57
557	5	57
558	5	58
1-10 of 1,000 rows	Pr	revious 1 2 3 4 5 100 Next

Logs: log(), log2(), log10(). Logarithms are an incredibly useful transformation for dealing with data that ranges across multiple orders of magnitude. They also convert multiplicative relationships to additive, a feature we'll come back to in modelling.

All else being equal, I recommend using | log2 () | because it's easy to interpret: a difference of 1 on the log scale corresponds to doubling on the original scale and a difference of -1 corresponds to halving.

• Offsets: lead() and lag() allow you to refer to leading or lagging values. This allows you to compute running differences (e.g. x - lag(x)) or find when values change (|x != lag(x)). They are most useful in conjunction with |group_by()|, which you'll learn about shortly.

Exercises

√ Exercise 1

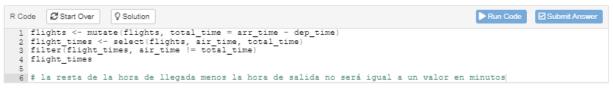
Currently dep_time and sched_dep_time are convenient to look at, but hard to compute with because they're not really continuous numbers. Convert them to a more convenient representation of number of minutes since midnight.

dep_tir <ir< th=""><th></th><th></th><th>dep_time <int></int></th><th>sched_dep_mins <dbl></dbl></th></ir<>			dep_time <int></int>	sched_dep_mins <dbl></dbl>
5	17 3	17	515	315
5	33 3	33	529	329
5	42 3	42	540	340
5	44 3	44	545	345
5	54 3	54	600	360
5	54 3	54	558	358
5	55 3	55	600	360
5	57 3	57	600	360
5	57 3	57	600	360
5	58 3	58	600	360
1-10 of 1,000 rows			Previous 1 2 3	4 5 6 100 Next

Good Job!

√ Exercise 2

Compare air_time with arr_time - dep_time . What do you expect to see? What do you see? How do you explain this?



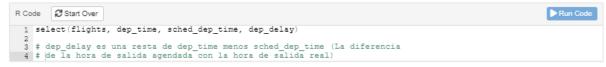
air_time <dbl></dbl>	total_time <int></int>
227	313
227	317
160	381
183	460
116	258
150	186
158	358
53	152
140	281
138	195
1-10 of 1,000 rows	Previous 1 2 3 4 5 6 100 Next

air_time <dbl></dbl>	total_time <int></int>
227	313
227	317
160	381
183	460
116	258
150	186
158	358
53	152
140	281
138	195
1-10 of 1,000 rows	Previous 1 2 3 4 5 6 100 Next

Good Job! it doesn't make sense to do math with `arr_time` and `dep_time` until you convert the values to minutes past midnight (as you did with `dep time` and `sched dep time` in the previous exercise).

✓ Exercise 3

Compare dep_time, sched_dep_time, and dep_delay. How would you expect those three numbers to be related?



dep_time <int></int>	sched_dep_time <int></int>	dep_delay <dbl></dbl>
517	515	2
533	529	4
542	540	2
544	545	-1
554	600	-6
554	558	-4
555	600	-5
557	600	-3
557	600	-3
558	600	-2
1-10 of 1,000 rows	Previous 1 2 3 4	5 6 100 Next

Exercise 4

Find the 10 most delayed flights (dep_delay) using a ranking function. How do you want to handle ties? Carefully read the documentation for min_rank().



year <int></int>		d <int></int>	dep_time <int></int>	sched_dep_time <int></int>	dep_delay <dbl></dbl>	arr_time <int></int>	sched_arr_time <int></int>	arr_delay <dbl></dbl>		,
2013	1	9	641	900	1301	1242	1530	1272	НА	
2013	6	15	1432	1935	1137	1607	2120	1127	MQ	
2013	1	10	1121	1635	1126	1239	1810	1109	MQ	
2013	9	20	1139	1845	1014	1457	2210	1007	AA	
2013	7	22	845	1600	1005	1044	1815	989	MQ	
2013	4	10	1100	1900	960	1342	2211	931	DL	
2013	3	17	2321	810	911	135	1020	915	DL	

√ Exercise 5

What does 1:3 + 1:10 return? Why?

```
R Code Start Over  Hint

1 1:3 + 1:10
2 3  # Al terminar la longitud de 1:3, repite el conjunto de datos para los demás valores de 1:10

Warning in 1:3 + 1:10: longer object length is not a multiple of shorter object length

[1] 2 4 6 5 7 9 8 10 12 11

Nicel R repeats 1:3 three times to create a vector long enough to add to 1:10. Since the length of the new vector is not exactly the length of 1:10, R also returns a warning message.
```

√ Exercise 6

What trigonometric functions does R provide? Hint: look up the help page for Trig.



Previous Topic