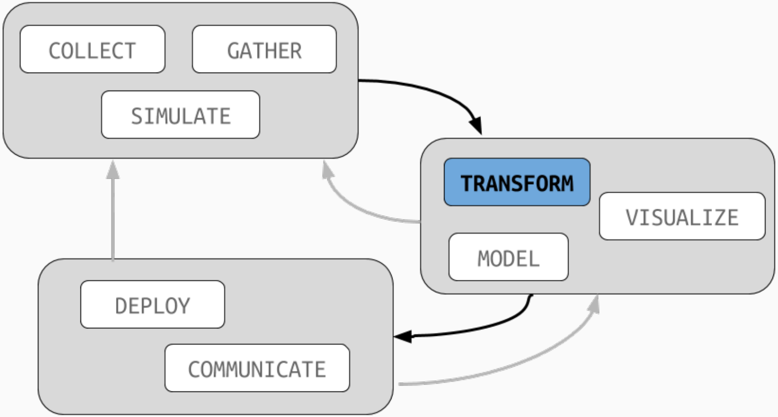
Lecture 21: Manipulating character vectors

Taylor Arnold


COLLECT
GATHER
SIMULATE
TRANSFORM
VISUALIZE
MODEL
DEPLOY
COMMUNICATE

relabeling categories

Notice that the season and weather conditions in the bikes dataset are labeled as integers:

```
select(bikes, season, weather)
```

```
## # A tibble: 731 x 2
##   season weather
##    <int>   <int>
## 1      1       2
## 2      1       2
## 3      1       1
## 4      1       1
## 5      1       1
## 6      1       1
## # ... with 725 more rows
```

relabeling categories

What if we wanted to create a new variable that had these properly labeled? I showed an example of doing this with the `left_join` function when I first introduced it. An easier way, at least in many cases, is to manually change the values one by one.

We have primarily used the `mutate` function for changing values in a dataset. However, the downside is that mutate requires us to change all the values at once. It is possible to do this using the `ifelse` function, but there is an easier way.

relabeling categories

Here we use base R functions to modify the dataset in place:

```
bikes$season_name <- "missing"
bikes$season_name[bikes$season == 1] <- "Winter"
bikes$season_name[bikes$season == 1] <- "Spring"
bikes$season_name[bikes$season == 1] <- "Summer"
bikes$season_name[bikes$season == 1] <- "Fall"
```

relabeling categories

It may look like this requires a lot of code, but its actually very little if you make good use of copy and paste.

This relabeling is very useful, and could be used by almost all of the datasets at some point. It can also be used to change labels that we already have or to create hand-constructed buckets.