Day 3: Data wrangling practice: teacher key

Casey O'Hara

2023-05-08

```
### Set message and warning to FALSE to suppress warnings and messages during
### package loading (e.g, library(tidyverse))
knitr::opts_chunk$set(echo = TRUE, message = FALSE, warning = FALSE)
```

Tour of R project:

- So far, just a file with a .Rproj extension. This file indicates "home base" for this project - the root directory.
- If we move this project directory somewhere else on our hard drive, that .Rproj goes with it.
- We can use that to our advantage to ensure all file paths are coded relative to that file.
- The alternative (absolute file paths) tend to point to your specific computer's name and directory system, which will not match anyone else's.

Tour of R Markdown:

- Markdown chunks:
  - easily format text using shortcuts to HTML tags.
  - These chunks are useful to write notes, comments, and explanations about what you are intending to do with your code.
- code chunks:
  - shaded differently for ease of spotting them
  - start with a "fence" - three backticks, then a curly-brace header with info about the chunk, including language, label, and chunk options.
  - write all your R code - objects will stay in memory even if you leave this chunk and go to another.
  - end with another fence.
  - can type manually, or shortcuts or menu.

Let's load some packages! Note: not a bad idea to do in setup chunk, but make sure to set up code chunk headers correctly!

```
library(poLCA) ### LCA package with some more data we'll use
library(tidyverse)
  ### dialect of R - cleans up and standardizes a lot of wrangling and data vis.
  ### note the packages listed in the message!

library(here) ### helps with creating relative pathways relative to .Rproj
library(palmerpenguins) ### some data we'll explore
```

At this point, knit the document to create a .html - note the warnings and errors - go back and adjust the setup chunk to avoid this.

Tidyverse overview

what is `tidyverse`?

The `tidyverse` package is a metapackage containing multiple other packages that have various uses for data wrangling, analysis, and visualization. The main ones we care about are:

- `readr` - fast and efficient reading in of data from CSVs and other tabular formats
- `dplyr` - managing, modifying, and working with data frames - 95% of our data wrangling toolbox
- `tidyr` - swapping between tabular formats - wide vs. long
- `ggplot2` - data visualization
- `forcats` - working with categorical variables in dataframes

Others you may be interested in:

- `stringr` - working with string/character data
- `purrr` - iteration
- `lubridate` - working with date formatted data
- `tidymodels` - consistent modeling across multiple model types

Base R is totally fine too, but the syntax is often idiosyncratic and/or hard to interpret. The `tidyverse` packages were designed to smooth out inconsistencies, improve readability, and in many cases boost performance. A good data scientist would want to be proficient in both dialects (and maybe `data.table`) as well as other languages.

Tidy data principles

- Each variable forms a column.
- Each observation forms a row.
- Each type of observational unit forms a table

counter examples:

- Column headers are values, not variable names.
- Multiple variables are stored in one column.
- Variables are stored in both rows and columns.
- Multiple types of observational units are stored in the same table.
- A single observational unit is stored in multiple tables.

year	artist	track	time	date.entered	wk1	wk2	wk3
2000	2 Pac	Baby Don't Cry	4:22	2000-02-26	87	82	72
2000	2Ge+her	The Hardest Part Of ...	3:15	2000-09-02	91	87	92
2000	3 Doors Down	Kryptonite	3:53	2000-04-08	81	70	68
2000	98^0	Give Me Just One Nig...	3:24	2000-08-19	51	39	34
2000	A*Teens	Dancing Queen	3:44	2000-07-08	97	97	96
2000	Aaliyah	I Don't Wanna	4:15	2000-01-29	84	62	51
2000	Aaliyah	Try Again	4:03	2000-03-18	59	53	38
2000	Adams, Yolanda	Open My Heart	5:30	2000-08-26	76	76	74

Table 7: The first eight Billboard top hits for 2000. Other columns not shown are `wk4`, `wk5`, ..., `wk75`.

year	artist	time	track	date	week	rank
2000	2 Pac	4:22	Baby Don't Cry	2000-02-26	1	87
2000	2 Pac	4:22	Baby Don't Cry	2000-03-04	2	82
2000	2 Pac	4:22	Baby Don't Cry	2000-03-11	3	72
2000	2 Pac	4:22	Baby Don't Cry	2000-03-18	4	77
2000	2 Pac	4:22	Baby Don't Cry	2000-03-25	5	87
2000	2 Pac	4:22	Baby Don't Cry	2000-04-01	6	94
2000	2 Pac	4:22	Baby Don't Cry	2000-04-08	7	99
2000	2Ge+her	3:15	The Hardest Part Of ...	2000-09-02	1	91
2000	2Ge+her	3:15	The Hardest Part Of ...	2000-09-09	2	87
2000	2Ge+her	3:15	The Hardest Part Of ...	2000-09-16	3	92
2000	3 Doors Down	3:53	Kryptonite	2000-04-08	1	81
2000	3 Doors Down	3:53	Kryptonite	2000-04-15	2	70
2000	3 Doors Down	3:53	Kryptonite	2000-04-22	3	68
2000	3 Doors Down	3:53	Kryptonite	2000-04-29	4	67
2000	3 Doors Down	3:53	Kryptonite	2000-05-06	5	66

Table 8: First fifteen rows of the tidied billboard dataset. The `date` column does not appear in the original table, but can be computed from `date.entered` and `week`.

Tidy wrangling in R

- Dataframe as main data type - columns are vectors of a particular class
- `mutate`, `filter`, `select`, `group_by`/`summarize` are the key dplyr functions in a data wrangling workflow
- Pipe operator `%>%` (or `|>` now native to R) for communicating flow - also vertical organization, spacing, etc to make it easy to read
- `pivot_wider` and `pivot_longer` (older: `spread` and `gather`) are key `tidyr` functions to go from wide to long and back, depending on needs of a modeling function

Enough chitchat, let's get to it

Several use cases to explore, focusing on working with discrete variables:

- continuous variable but want to turn it into discrete, e.g., binning income levels

  - binary vs. nominal vs. ordinal (or dichotomous vs. polytomous)
  - `cut()`, `ntile()`, `ifelse()`, and `case_when()`
- multi-value discretes as factors - ordered or unordered
- spreading multi-valued discretes into multiple dummies

Here we will:

- Load a dataset built into an R package and inspect it
- Convert some columns into categoricals
- Run a simple linear regression to see how R handles categoricals

```
penguins <- palmerpenguins::penguins
### NOT normal way to load data - usually reading in a CSV
### could also do data(penguins)

summary(penguins)
```

```
##       species          island    bill_length_mm  bill_depth_mm
##  Adelie   :152   Biscoe   :168   Min.   :32.10   Min.   :13.10
##  Chinstrap: 68   Dream    :124   1st Qu.:39.23   1st Qu.:15.60
##  Gentoo   :124   Torgersen: 52   Median :44.45   Median :17.30
##                                  Mean   :43.92   Mean   :17.15
##                                  3rd Qu.:48.50   3rd Qu.:18.70
##                                  Max.   :59.60   Max.   :21.50
##                                  NA's   :2       NA's   :2
##  flipper_length_mm  body_mass_g       sex           year
##  Min.   :172.0     Min.   :2700   female:165   Min.   :2007
##  1st Qu.:190.0     1st Qu.:3550   male  :168   1st Qu.:2007
##  Median :197.0     Median :4050   NA's  : 11   Median :2008
##  Mean   :200.9     Mean   :4202                Mean   :2008
##  3rd Qu.:213.0     3rd Qu.:4750                3rd Qu.:2009
##  Max.   :231.0     Max.   :6300                Max.   :2009
##  NA's   :2         NA's   :2
```

```
glimpse(penguins)
```

```
## Rows: 344
## Columns: 8
## $ species           <fct> Adelie, Adelie, Adelie, Adelie, Adelie, Adelie, Adel~
## $ island            <fct> Torgersen, Torgersen, Torgersen, Torgersen, Torgerse~
## $ bill_length_mm    <dbl> 39.1, 39.5, 40.3, NA, 36.7, 39.3, 38.9, 39.2, 34.1, ~
## $ bill_depth_mm     <dbl> 18.7, 17.4, 18.0, NA, 19.3, 20.6, 17.8, 19.6, 18.1, ~
## $ flipper_length_mm <int> 181, 186, 195, NA, 193, 190, 181, 195, 193, 190, 186~
## $ body_mass_g       <int> 3750, 3800, 3250, NA, 3450, 3650, 3625, 4675, 3475, ~
## $ sex               <fct> male, female, female, NA, female, male, female, male~
## $ year              <int> 2007, 2007, 2007, 2007, 2007, 2007, 2007, 2007, 2007~
```

```
head(penguins)
```

```
## # A tibble: 6 x 8
##   species island    bill_length_mm bill_depth_mm flipper_l~1 body_~2 sex    year
```

```
##   <fct>   <fct>              <dbl>         <dbl>       <int>   <int> <fct> <int>
## 1 Adelie  Torgersen           39.1          18.7         181    3750 male   2007
## 2 Adelie  Torgersen           39.5          17.4         186    3800 fema~  2007
## 3 Adelie  Torgersen           40.3          18           195    3250 fema~  2007
## 4 Adelie  Torgersen           NA            NA            NA      NA <NA>   2007
## 5 Adelie  Torgersen           36.7          19.3         193    3450 fema~  2007
## 6 Adelie  Torgersen           39.3          20.6         190    3650 male   2007
## # ... with abbreviated variable names 1: flipper_length_mm, 2: body_mass_g
```

```
peng_subset <- penguins %>%
  select(species, bill_depth_mm, body_mass_g) %>%
  filter(species %in% c('Adelie', 'Gentoo'))

peng_sum <- penguins %>%
  group_by(species) %>%
  mutate(mass_normalized = body_mass_g / max(body_mass_g, na.rm = TRUE)) %>%
  summarize(mean_mass = mean(body_mass_g, na.rm = TRUE),
            mean_norm = mean(mass_normalized, na.rm = TRUE),
            bill_ratio = mean(bill_length_mm / bill_depth_mm, na.rm = TRUE))

peng_sum
```

```
## # A tibble: 3 x 4
##   species   mean_mass mean_norm bill_ratio
##   <fct>         <dbl>     <dbl>      <dbl>
## 1 Adelie        3701.     0.775       2.12
## 2 Chinstrap     3733.     0.778       2.65
## 3 Gentoo        5076.     0.806       3.18
```

```
### Create new categorical columns from existing numerics
peng_cats <- penguins %>%
  select(-island, -sex, -year) %>%
  drop_na() %>%
  rename(bl = bill_length_mm,
         bd = bill_depth_mm,
         fl = flipper_length_mm,
         bm = body_mass_g) %>%
  ### ifelse for binary
  mutate(bl_cat = ifelse(bl > 43.92, 'long', 'short')) %>%
  # mutate(bl_cat = ifelse(bl > mean(bl, na.rm = TRUE), 'long', 'short')) %>%
  ### logical test for binary
  mutate(bl_long_lgl = (bl > 43.92)) %>%
  # mutate(bl_long_lgl = (bl > median(bl, na.rm = TRUE))) %>%
  ### case_when for multiple assigned values
  mutate(bd_cat = case_when(bd < 15.6 ~ 'shallow',
                            bd > 18.7 ~ 'deep',
                            TRUE ~ 'medium')) %>%
  ### ntile for equal-sized groups
  mutate(fl_quartile = ntile(fl, 4),            ### careful of handling NAs!
         fl_quartile = factor(fl_quartile)) %>% ### convert numeric to factor
  ### cut for equal-sized bins
  mutate(bm_cut = cut(bm, 5)) ### already a factor

head(peng_cats)
```

```
## # A tibble: 6 x 10
##   species    bl    bd    fl    bm bl_cat bl_long_lgl bd_cat fl_quartile bm_cut
##   <fct>   <dbl> <dbl> <int> <int> <chr>  <lgl>       <chr>  <fct>       <fct>
## 1 Adelie   39.1  18.7   181  3750 short  FALSE       medium 1           (3.42e+~
## 2 Adelie   39.5  17.4   186  3800 short  FALSE       medium 1           (3.42e+~
## 3 Adelie   40.3  18     195  3250 short  FALSE       medium 2           (2.7e+0~
## 4 Adelie   36.7  19.3   193  3450 short  FALSE       deep   2           (3.42e+~
## 5 Adelie   39.3  20.6   190  3650 short  FALSE       deep   1           (3.42e+~
## 6 Adelie   38.9  17.8   181  3625 short  FALSE       medium 1           (3.42e+~
```

Left off here for Day 3 synchronous session - run the above then continue from here!

Try out some categorical values in context of a linear model to see how R interprets things!

```
### examine a linear model of body mass as a function of a combo of
### different numeric and categorical variables
peng_lm <- lm(bm ~ bl + bd_cat + fl_quartile + species, data = peng_cats)

summary(peng_lm) ### note reference values and effect of nonref values
```

```
##
## Call:
## lm(formula = bm ~ bl + bd_cat + fl_quartile + species, data = peng_cats)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -794.62 -230.02  -15.09  197.55 1144.69
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       1627.580    285.666   5.698 2.68e-08 ***
## bl                  55.467      7.206   7.698 1.59e-13 ***
## bd_catmedium      -251.507     50.549  -4.976 1.04e-06 ***
## bd_catshallow     -586.796     90.705  -6.469 3.52e-10 ***
## fl_quartile2       140.727     53.536   2.629 0.008970 **
## fl_quartile3       275.049     71.647   3.839 0.000148 ***
## fl_quartile4       571.496    102.837   5.557 5.62e-08 ***
## speciesChinstrap -630.373     84.880  -7.427 9.42e-13 ***
## speciesGentoo      814.049    125.563   6.483 3.24e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 333.8 on 333 degrees of freedom
## Multiple R-squared:  0.8308, Adjusted R-squared:  0.8267
## F-statistic: 204.4 on 8 and 333 DF,  p-value: < 2.2e-16
```

How to interpret those coefficients? (all stats sig, $p < .05$)

- effect of bill length on mass?

- effect of bill depth category? what is the reference value and why?
- effect of flipper length quartile? ref value and why?
- effect of species? reference value and why?

In this case, the `lm()` function takes a categorical variable, assigns one value as dummy (first level if factor OR first alphabetically if character), then auto creates dummies (T/F or 1/0 for each other level/value).

What if our modeling function required all dummy vars?

We can spread the various values of flipper length quartile (or bill depth category) into individual columns. This might be more common for variables with multiple, non-mutually-exclusive values (e.g., `diet` may include instances of vegetables, legumes, dairy, each of which might need to be split into a separate column of T/F).

```
peng_cats2 <- peng_cats %>%
  mutate(bd_val = TRUE) %>%
  ### try this first, note bd_val is consumed; and ask how to improve:
  # pivot_wider(names_from = bd_cat, values_from = bd_val)
  pivot_wider(names_from = bd_cat, values_from = bd_val,
              values_fill = FALSE, names_prefix = 'bd_') %>%
  ### ask students to do the same with another column -  e.g., species, island,
  ###   fl_quartile - and maybe use 1/0 or some other fill value
  mutate(fl_val = 1) %>%
  pivot_wider(names_from = fl_quartile, values_from = fl_val,
              values_fill = 0, names_prefix = 'fl_')

head(peng_cats2)
```

```
## # A tibble: 6 x 15
##   species    bl    bd    fl    bm bl_cat bl_lon~1 bm_cut bd_me~2 bd_deep bd_sh~3
##   <fct>   <dbl> <dbl> <int> <int> <chr>  <lgl>    <fct>  <lgl>   <lgl>   <lgl>
## 1 Adelie   39.1  18.7   181  3750 short  FALSE    (3.42~ TRUE    FALSE   FALSE
## 2 Adelie   39.5  17.4   186  3800 short  FALSE    (3.42~ TRUE    FALSE   FALSE
## 3 Adelie   40.3  18     195  3250 short  FALSE    (2.7e~ TRUE    FALSE   FALSE
## 4 Adelie   36.7  19.3   193  3450 short  FALSE    (3.42~ FALSE   TRUE    FALSE
## 5 Adelie   39.3  20.6   190  3650 short  FALSE    (3.42~ FALSE   TRUE    FALSE
## 6 Adelie   38.9  17.8   181  3625 short  FALSE    (3.42~ TRUE    FALSE   FALSE
## # ... with 4 more variables: fl_1 <dbl>, fl_2 <dbl>, fl_3 <dbl>, fl_4 <dbl>,
## #   and abbreviated variable names 1: bl_long_lgl, 2: bd_medium, 3: bd_shallow
```

Note here, each penguin has only one TRUE value across the bd_X columns, since bill depth bins are mutually exclusive. But if we were studying something like diet, perhaps there would be multiple columns of food items - krill, squid, crab, etc. - potentially with multiple TRUE observations across the various items since they are not mutually exclusive.

If our data were in a wide format but we wanted to turn it into a long format, we can use `pivot_longer()` to gather multiple columns into a key-value pair of columns. Note before pivot_longer we create a penguin_id column because we may have some observations with multiple TRUE values, in which case we'd have multiple rows for a single penguin. That's not tidy data! (but in this case, we only have one row per observation so we're OK)

```r
peng_cats3 <- peng_cats2 %>%
  mutate(penguin_id = 1:n()) %>% ### create identifier for each penguin observation
  pivot_longer(names_to = 'bd_cat', values_to = 'bd_val', starts_with('bd_')) %>%
  filter(bd_val == TRUE)

head(peng_cats3)
```

```
## # A tibble: 6 x 15
##   species    bl    bd    fl    bm bl_cat bl_lon~1 bm_cut  fl_1  fl_2  fl_3  fl_4
##   <fct>   <dbl> <dbl> <int> <int> <chr>  <lgl>    <fct>  <dbl> <dbl> <dbl> <dbl>
## 1 Adelie   39.1  18.7   181  3750 short  FALSE    (3.42~     1     0     0     0
## 2 Adelie   39.5  17.4   186  3800 short  FALSE    (3.42~     1     0     0     0
## 3 Adelie   40.3  18     195  3250 short  FALSE    (2.7e~     0     1     0     0
## 4 Adelie   36.7  19.3   193  3450 short  FALSE    (3.42~     0     1     0     0
## 5 Adelie   39.3  20.6   190  3650 short  FALSE    (3.42~     1     0     0     0
## 6 Adelie   38.9  17.8   181  3625 short  FALSE    (3.42~     1     0     0     0
## # ... with 3 more variables: penguin_id <int>, bd_cat <chr>, bd_val <lgl>, and
## #   abbreviated variable name 1: bl_long_lgl
```