Check-in 4: Answers

Again, the check-ins will review material presented in class but will also require you to think about new concepts, integrate across topics, and search for information. Some will be complex and take time to figure out. Feel free to work in groups on this.

Insert answers within the code chunks. Unless specified otherwise, do not assign your output to an object. If directed to assign output to an object, wrap the pipe in parentheses to print the output.

First, import `stevens_etal_2020_obed_data1.csv` from the following URL and save it in your `checkins` directory as `obed_data.csv`:

https://decisionslab.unl.edu/data/stevens_etal_2020_obed_data1.csv

Problem 1: (5 pt) Inside the code chunk below, write a **single command** (using pipes) that assigns to the object `dog_data` the following in this order:

1. Reads in `obed_data.csv`.
2. Includes only the follow variables in this order: `date`, `class`, `dog_sex`, `cgc_test`, all four `cort` columns.
3. Excludes observations where the dog's sex is missing.
4. Divides up the `date` variable into `year`, `month`, and `day` and keeps the `date` column.
5. Creates a new column called `session` that combines the year with the class separated by a slash `/` and keeps the `year` and `class` columns.

Problem 2: (3 pts) For `dog_data`, write a single command that does the following and assigns it to `dog_data2`:

1. Creates an `id` variable that is a sequence from 1 to the length of the data frame.
2. Makes the `id` variable the first column.
3. Reshapes the data frame to be long format such that the four `cort` columns are turned into a column of labels called `time` and a column of values called `cort`.

Problem 3: (2 pts) Looks like more data have come in. After defining `new_data` in the code chunk below, append these new data to the bottom of `dog_data` and sort by *reverse* chronological order by date.

```r
new_data <- tibble(date = as.Date("2021-05-18"), class = "U21", dog_sex = "Female",
  cgc_test = "Pass", cort1 = 0.254, cort2 = NA, cort3 = NA, cort4 = 0.188)
```

Problem 4: (2 pts) The location data for each class was saved in a different data frame. After defining `location` in the code chunk below, merge the `location` data frame with `dog_data` based on `class`/`semester` and then move the `site` column after the `class` column.

```
location <- tibble(semester = c("U18", "U19", "S19", "F18", "S18", "F19"),
  site = c("south", "south", "campus", "campus", "campus", "campus"))
```

Problem 5: (2 pts) Do the following:

1. Create an object called `campus` that keeps rows of `location` in which the site is "campus".
2. Use a filtering join to keep rows of `dog_data` that match the semester found in `campus`.

Problem 6: (3 pts) Do the following for `dog_data`:

1. Find **all possible** combinations of `class`, `dog_sex`, and `cgc_test` (ignoring NAs).
2. Find **all existing** combinations of `class`, `dog_sex`, and `cgc_test` (ignoring NAs).
3. Find the possible combinations of `class`, `dog_sex`, and `cgc_test` that **do not exist** in the data set (ignoring NAs).

Problem 7: (3 pts) Do the following to `dog_data2` in a single pipeline:

1. Remove rows where `cort` is `NA`.
2. Insert all implicitly missing `NA`s.
3. Reshape the cort data to have a single row per `id` and the four `cort` values spread into four columns.

Problem 8: (3 pts) In your own creative way, I would like you to **extend** beyond what the previous questions have asked you to do. So build a pipeline with at least three steps that does at least one (preferably all) of the following (using this data set or another one):

1. Uses a function from this module
2. Uses a function from a previous module
3. Uses a function that we haven't covered in the course