Dirty Data Feedback: Vera

Reviewer: David

Positives

This was a well-written, well-worked project that delivered analysis well! Good use of R for Data Analysis throughout making use of the `{tidyverse}` suite of packages. Well done

Cleaning

Good approach and workflow to your cleaning:

- investigating data and removing unnecessary information using `names()` and `select()`
- recoding the country column
- cleaning the age column
- getting year from datestamp column
- unifying names before binding
- combining into one dataset

Good use of comments, coding style and whitespace to assist with readability of your work.

You made very good use of `{dplyr}`, `{stringr}`, and `%>%`s for data cleaning.

Good to see some researched functions and methods: `rename_with()` being a prime example. Great way to drop the Q6 from the 2017 data. Also good use of base R: `unique(data$column)` to grab the distinct values in a column.

Good checking of columns with `intersect()` and `set_diff()` before binding.

Analysis

Overall a good analysis document using good `{dplyr}`, tidy data principles, and data visualisation.

You consistently wrote readable code that correctly answered the questions and provided a meaningful commentary using the markdown section of your notebook. Well done.

Well documented cleaning steps.

Good use of `slice_max()` for grouped summary tables.

Answers were all correct.

Nice clear visualisations that effectively told a story of your data.

Potential Improvements

It's a good project that cleans, analyses and visualises the data well. Next steps I think are just to tidy up a bit of the formatting and formalise it as a project and part of your portfolio.

Cleaning

Your cleaning script was very good and prepared the data so that you could analyse it. My main next step for you would be to formalise it a bit:

- convert this into an R script (`cleaning.R` instead of `task_4_cleaning.Rmd`)
- reorder some parts
- remove checking functions

The idea behind a cleaning script is that it provides one document that can be run from top-to-bottom at once to read in raw data, clean it in some way, and then write the new clean data to a file. While it's normal practice to prepare this script in a Notebook (so you can check the flow of data at each part and make sure the cleaning is happening as you think). Once you've checked everything though, it's standard to convert this into a script.

some checking functions like `names()`, `ncol()`, `view()`, `intersect()`, setdiff()‘, etc. should be removed before formalising your cleaning into a script. Remember the idea of the cleaning script is that I just run it all at once if I need to, not that I run it line by line and then check the outputs. If it's important to check (maybe the next bit won't work if you have certain column names) then I'd use assertive programming instead to verify.

Remember that you can go back and change your order. Just because you bound the data before making the "going out" columns the same doesn't mean you can't go back to before you executed that part. This is preferred usually to then fixing the result.

Unifying column “are you trick or treating” (because we forgot to do it before;
as a result, we united three columns and cleaned values in them from NA)

Why not go back and do it before?

rather than using nested `if_else()` a single `case_match()` or `case_when()` is preferred style:

```
candy_2016_country_clean <- candy_2016_del_range_country %>%
  mutate(Country = case_when(
    Country == "United States" ~ "United States",
    Country == "United Kindom" ~ "United Kingdom",
    Country == "Canada" ~ "Canada",
    .default == "OTHER"
  )

# or this can be written more simply:
candy_2016_country_clean <- candy_2016_del_range_country %>%
  mutate(Country = if_else(
    # if country is in the set of countries leave it as is, otherwise call it OTHER
    Country  %in% c("United States", "United Kingdom", "Canada"), Country, "OTHER"
    )
  )
```

Analysis

Remember to use the `{here}` package when reading in data to a file that is not in the same folder as your `.Rproj` file! I wouldn't have actually been able to reproduce your work because of this.

```
library(here)
candy_analysis <- read_csv(here("clean_data/clean_candy.csv"))
```

Remember in code chunks: # is used to create a code comment. In the space outside code chunks # is used to define a large title/heading. So your document starts with about 20 large headings. You can write commentary in a notebook just by typing outside of a code chunk (no need for a #).

I'd turn your cleaning steps into a list:

1.
2.
3.

I'd mention something about data assumptions: e.g. when someone entered an invalid response did you remove all of their responses or just the invalid ones?

I'd sort indentation in a few places:

```
candy_totals <- candy_analysis_ratings %>%
      gather("candy_bar","ratings") %>%
        group_by(candy_bar, ratings) %>% count(ratings, name = "total") %>%
       arrange(desc(`total`))
head(filter(candy_totals, ratings == "DESPAIR"))
head(filter(candy_totals, ratings == "JOY"))
head(filter(candy_totals, ratings == "MEH"))
```

Instead of pivoting multiple times I would create one long dataset and adapt that for the questions that needed it.