Describing Data: Candy Wars

In this assignment, you will explore the data associated with the FiveThirtyEight story The Ultimate Halloween Candy Power Ranking. There are three parts to the assignment. Parts 1 and 2 must be submitted according to their specifications to earn a B (3.2) on the assignment. You may choose to submit Part 3 to earn an A on the assignment – be aware that you can only earn an A by completing the B-level components first (i.e., you shouldn't do this part unless your Part 1 and Part 2 are complete and correct). Each part provides its own set of instructions. The general specifications for this assignment, which are required to achieve a passing grade, are:

- Answers to questions from all parts submitted as a single .pdf, images submitted as .png files exported from code (no screenshots) with names as indicated, code for all sections submitted as a single .py or .ipynb file.
- .pdf questions file and code file labeled clearly throughout as to which part corresponds to which question/task.
- Files submitted are readable in Blackboard (except .ipynb files)– this means you must check your submission after turning it in.
- Free of spelling, capitalization, and large-scale grammatical errors.
- For a group submission: only one submission in Blackboard by one group member with other members listed in the submission comments (if you don't know how to do this, please ask).
- Work is your own/your group's own, with sources cited – class notes and official Python documentation do not need to be cited; any unauthorized collaboration/sources or use of allowed sources without citation invalidates the submission.

The instructions that follow are lengthy. They tell you exactly what to do, so read them carefully and consult them as you complete your work. I would suggest using the specifications as a checklist – go over them several times to make sure you have met them in full.

Part 1

Question: Does chocolate reign supreme?

For the article, The Ultimate Halloween Candy Power Ranking, FiveThirtyEight compiled data from an online survey where participants were asked which of two candies they preferred. You can find a description of the original data and how it was collected here: https://github.com/fivethirtyeight/data/tree/master/candy-power-ranking. In this part of the assignment, you will be working with a slightly altered version of the data provided in the file `Candy Data.csv` – use this data file – do not download the original data.

We are first interested in exploring whether chocolate candy is consistently preferred to non-chocolate candy.

One of the variables in your data categorizes how frequently a candy wins. The encoding of `win_cat` is: `a_lot` (wins >= 75%), `often` (75% > wins >= 50%), `sometimes` (50% > wins >= 25%), `rarely` (wins < 25%). A second variable categorizes candy by type. The encoding of `is_chocolate` is: `chocolate`, `fruity`, `both`, `neither`.

There is only one item that is marked as `both`, the `Tootsie Roll Pop`. Before you create the visualization described below, create a new data frame that has this observation removed. IMPORTANT: Subset the data and create a NEW data frame – do not alter the original one. If you're unsure how to do this, ask for help. This means you should end up with two variables that correspond to data frames, one with the observation and one without the observation.

Task 1:

Create a stacked bar chart using the data frame that excludes the `Tootsie Roll Pop` and summarizes the proportion of each winning class that corresponds to each type of candy. This means you will group the data by the categorical win frequency and, within these groups, show the relative percentages of each type of candy.

The relevant columns in the data set, as described above, are:

```
win_cat
is_chocolate
```

The specifications for the visualization (i.e., what you have to adhere to to earn full credit) are given below. Some tips on how to proceed are also included. Read these carefully and review them multiple times; they will tell you exactly what to do and what the instructor will be looking for when evaluating the assignment.

A visualization that receives full credit will:

- Be a stacked bar chart as described above, with categories displayed as described; use the guidelines (and code!) for stacked bar charts from class as opposed to default settings.
- Exclude the single observation that has the value `both` for `is_chocolate`; add a text box under/near the legend that contains the text: 'Tootsie Roll Pops, which are both Fruity and Chocolate, were excluded from the figure'; the text box should have a background color that is different from the regular background; add in line breaks so that the text does not appear all on one line (ask if you do not understand this) and center the text.
- Select colors for the plot based on the principles discussed in class and the reading; you can see a complete list of named colors in matplotlib here: https://matplotlib.org/3.1.0/gallery/color/named_colors.html.
- Re-format all labels, those under the bars and in the legend, to contain the size of each group. For example, we would have labels like: 'A Lot (5)' and 'Chocolate (36)'.
- Label the values inside the bars as was done in class.
- Use labels (axis labels, data labels, legend items, etc.) that are in title case and of appropriate font size as per our discussions in class and the readings; be sure to include additional labeling and remove default labeling as needed.
- Place the legend such that it is fully legible, and re-format legend components to maximize readability and minimize non-data ink; the legend must contain the correct information; the legend title should not just be the variable name and should be appropriately formatted (title case, centered) as described above.
- Ensure that your figure is large enough that all components are visible and identifiable.
- Save the figure in .png format as `stacked_bar.png`; save your plot to the file using `plt.savefig('name.png', bbox_inches = 'tight', facecolor = 'white');` check that your files are saved correctly. If you are getting blank figures, you may need a different code – ask for help!
- Adhere to guidelines in class for readability (e.g., removing non-data ink), truthfulness (no data hiding or distortion), and effectiveness.
- Provide Python code used to generate the plot; code will run without errors to re-generate the plot; code uses matplotlib and seaborn libraries for plotting.

Task 2

Answer the following questions in a few sentences (2-3 is fine. Longer is ok as well) as described below:

A. Why are there two variables in the data that relate to the win percentage? Why did we use the one we did to make this figure (i.e., why didn't we use the other one)?

B. What do you determine about frequent winners versus less frequent winners? Your answer should ONLY be based on your visualizations and must be faithful to your visualizations. Do not include outside information that is not contained in the data or the visualization.

Part 2

Question: How do chocolate and non-chocolate candies differ in winningness and price?

We will look at the overall distributions of win frequency and price for chocolate and non-chocolate candy using histograms. The relevant variables in the data set are:

```
winpercent
pricepercent
is_chocolate
```

The `pricepercent` variable isn't the actual price – it is a normalized value between 0 and 1, reflecting where the price falls in the overall collection of prices. Candy considered chocolate will have the value `chocolate` for `is_chocolate`. Non-chocolate candy is all candy with any other value (i.e., anything BUT `chocolate`).

Make sure you use the ORIGINAL data (not the data frame with the one observation removed).

Task 1:

Create a set of histograms as subplots of one larger plot that show the distributions of `winpercent` and `pricepercent` for chocolate and non-chocolate candy. This means you should have four histograms – do not create overlapping histograms. Create four separate subplots. These should be organized in a 2x2 grid, which means two rows of 2 plots each.

The specifications for the visualization (i.e., what you have to adhere to to earn full credit) are given below. Some tips on how to proceed are also included. Read these carefully and review them multiple times; they will tell you exactly what to do and what the instructor will be looking for when evaluating the assignment.

A visualization that receives full credit will:

- Be a figure consisting of 4 subplots that contain histograms organized in a 2x2 grid as described above – you MUST create subplots. Place the two histograms for chocolate candy in the first row and the two histograms for non-chocolate candy in the second row.
- Contain histograms that do not use the defaults but rather show enhancements and customization based on material learned in class.
- Include all observations, including the single observation that has the value for `is_chocolate` as `both`.
- Select colors for the plot based on the principles discussed in class and the reading; you can see a complete list of named colors in matplotlib here: https://matplotlib.org/3.1.0/gallery/color/named_colors.html.
- Present the data truthfully by having axes that all have the same lower and upper limits.

- Standardize bar size or come close to it; I used bin sizes of 8 for `win_percent` and 15 for `price_percent` to achieve this.
- Label axes as needed, including removing extraneous labeling; you will find labeling MUCH easier if you do not `sharex` or `sharey`, instead set the plot limits manually using `xlim` and `ylim`.
- Use labels (axis labels, data labels, legend items, etc.) that are in title case and of appropriate font size as per our discussions in class and the readings; be sure to include additional labeling and remove default labeling as needed.
- May or may not use a legend depending on axis labels and other labeling. If you use a legend, place the legend such that it is fully legible, and re-format legend components to maximize readability and minimize non-data ink; the legend must contain the correct information; the legend title should not just be the variable name and should be appropriately formatted (title case, centered) as described above.
- Ensure that your figure is large enough that all components are visible and identifiable.
- Save the figure in .png format as `subplots.png`; save your plot to the file using `plt.savefig('name.png', bbox_inches = 'tight', facecolor = 'white');` check that your files are saved correctly. If you are getting blank figures, you may need a different code – ask for help!
- Adhere to guidelines in class for readability (e.g., removing non-data ink), truthfulness (no data hiding or distortion), and effectiveness.
- Provide Python code used to generate the plot; code will run without errors to re-generate the plot; code uses matplotlib and seaborn libraries for plotting.

Task 2

Answer the following question in a few sentences according to the instructions below:

> What are the noticeable differences in the winning percent and price percent distributions between chocolate and non-chocolate candy? Describe the trends you see in detail based on your visualization.

Part 3: This section is only required if you are seeking to earn an A on the assignment. You should only attempt this if you have completed the mandatory section. There are two options for this section; you only need to complete ONE of them, but you are welcome to do both if you'd like.

Option 1: How many of each type of candy were in the original survey?

We included sample sizes in our bar chart for each type of candy, but now we will display this graphically using a waffle plot. Read the instructions below very carefully and use them as a guide while you make the plot. They are precise.

You must use *pywaffle* to make the plot; be sure to use the original data that includes ALL of the observations. The first example on this page is a good place to start: https://pywaffle.readthedocs.io/en/latest/examples/plot_with_characters_or_icons.html. You can find proper documentation in general here: https://pywaffle.readthedocs.io/en/latest/.

Use symbols, specifically emoji, to represent each type of candy; the Font Awesome icons don't include icons that are necessarily the best for this task. Instead, use NotoEmoji (https://fonts.google.com/noto/specimen/Noto+Emoji) – the file you need has been provided on Blackboard, but you can download a different variation if you'd like. To use the file, include the argument `font_file` in your call to plt, like this:

```
font_file = 'where_you_put_the_file/NotoEmoji-VariableFont_wght.ttf'
```

Use the `characters` argument (NOT icons) and specify a different emoji for each of the four groups (`chocolate, not chocolate, both, neither`). Select a different color for each group as well. Include a legend that shows the name of each group and the color – you do NOT need to have the emoji in the legend (that is pretty hard to do).

A visualization that receives full credit will (read this; it has major hints!):

- Be a waffle plot made using *pywaffle*; use the ORIGINAL data, including all candy from the survey; you will find it easiest to calculate the totals beforehand and use those, i.e., hard code them or store them in some kind of list or dictionary – trying to pass a data frame into the plotting code most likely will not work; include your code – it must run without errors.
- Use emoji as described above, selecting emojis that represent the actual categories as best as possible, not randomly selected.
- Select colors based on the principles from class, and to enhance the emoji and create contrast.
- Include a legend for the colors as described above, which has careful consideration for font size and shows names in title case; consider legend placement carefully (hint: you can find information on how to create and format the legend in the PyWaffle documentation).
- Have a title that is in an appropriate font size and is well-placed in relation to the plot.
- Save the figure in .png format as `waffle.png` (make sure to use `bbox_inches` and `face_color`)

Option 2: Who were the ultimate winners? How do they compare?

We looked at the distribution of win percentages for chocolate versus non-chocolate in histograms in Part 2. Histograms group the data, so we will use a different type of plot to see each observation individually. Create a swarmplot using seaborn (https://seaborn.pydata.org/generated/seaborn.swarmplot.html) that displays the win percentage (on x) for individual candies grouped by type: `chocolate`, `fruity`, `neither`, and `both` (categories on y). Use colors to distinguish between `bar`, `pluribus`, and `neither`. You will also add text to identify the top winner in the `chocolate`, `fruity`, and `neither` categories.

To create a plot that receives full credit:

- Include the data required (`winpercent`), organized as needed (by category), in the swarmplot.
- Plot the swarmplot with a horizontal orientation (i.e., the lines of dots are horizontal).
- Add vertical grid lines with a non-distracting line style, color, and weight.
- Choose a marker size to optimize the visibility of the data.
- Use three different well-chosen colors (per Muth) to distinguish points that correspond to bar versus pluribus versus neither.
- Re-format all labels, including those in the legend, to be in title case and of appropriate size as per our discussions in class and the readings.
- Place the legend, which explains the colors used, such that it is fully legible, and re-format legend components to maximize readability and minimize non-data ink.
- Add legible text labels to specify the specific candy (e.g., M&M's – that's not an actual label you will have) that had the highest win percent for the chocolate, fruity, and neither categories; these labels should be placed to indicate the point they refer to; you can add arrows if you'd like to.
- Ensure that your figure is large enough that all components are visible and identifiable.
- Save the figure in .png format as `swarmplot.png`.
- Adhere to guidelines in class for readability, truthfulness, and effectiveness.
- Provide Python code used to generate the plot; code will run without errors to re-generate the plot; code uses seaborn to make the plot (you will also need matplotlib) – it does not make use of other plotting libraries.