Problem Set 6 - Waze Shiny Dashboard (ver. 1.1)

Peter Ganong, Maggie Shi, and Andre Oviedo

2024-11-21

Steps to submit (10 points on PS6)

1. "This submission is my work alone and complies with the 30538 integrity policy." Add your initials to indicate your agreement: **___**
2. "I have uploaded the names of anyone I worked with on the problem set **here**" **___** (2 point)
3. Late coins used this pset: **___** Late coins left after submission: **___**
4. Before starting the problem set, make sure to read and agree to the terms of data usage for the Waze data here.
5. Knit your `ps6.qmd` as a pdf document and name it `ps6.pdf`.
6. Submit your `ps6.qmd`, `ps6.pdf`, `requirements.txt`, and all created folders (we will create three Shiny apps so you will have at least three additional folders) to the gradescope repo assignment (5 points).
7. Submit `ps6.pdf` and also link your Github repo via Gradescope (5 points)
8. Tag your submission in Gradescope. For the Code Style part (10 points) please tag the whole correspondingsection for the code style rubric.

Notes: see the Quarto documentation (link) for directions on inserting images into your knitted document.

IMPORTANT: For the App portion of the PS, in case you can not arrive to the expected functional dashboard we will need to take a look at your ***`app.py`*** *file. You can use the following code chunk template to "import" and print the content of that file. Please, don't forget to also tag the corresponding code chunk as part of your submission!* # Background {-}

Data Download and Exploration (20 points)

Prior to starting the problem set, you should have downloaded the required data for this problem from here. The data dictionary for this dataset is here.

1. Using the `zipfile` package, unzip the `waze_data.zip` file. You will find two files in the unzipped folder: `waze_data.csv` (the whole dataset) and `waze_data_sample.csv` (a sample of 1% of the data). Load the `waze_data_sample.csv` file into a pandas DataFrame. What are the variable names and what are their data types? When reporting data types, report using the Altair syntax (e.g., Quantitative, Nominal, etc.). When reporting data types, ignore the columns `ts`, `geo`, and `geoWKT`.
2. Now load the `waze_data.csv` file into a pandas DataFrame. With this file, Create a stacked bar chart where the x-axis is each variable and the stacked bar has two categories: the number of observations where that variable is `NULL` or missing, and the number of observations where they are not. Which variables have the `NULL` values? Which variable has the highest share of observations that are missing?
3. Take a look at the variables `type` and `subtype`. Even though they informative, some are not aesthetically pleasing, and others are difficult to read. Before going into the development of our Shiny Apps, we will create a **crosswalk** table to help us have cleaner data.

    a. Print the unique values for the columns `type` and `subtype`. How many types have a subtype that is `NA`? Even though we print the combinations for two columns, can you identify which `type` has `subtypes` that have enough information to consider that they could have *sub-subtypes*?

    b. Write out a bulleted listed with the values at each layer given this hierarchy. For this list, use names that are clean and readable. For example, using `ACCIDENT_MAJOR` in the dashboard is not as readable or user-friendly as one menu option that says `Accident` and then a subsequent one that says `Major`.

    c. Finally, do you consider that we should keep the `NA` subtypes? Why? If you choose to keep the `NA` subtypes, code them as "Unclassified."

4. We want to assign this newly created hierarchy to the original data. To do so, we will create the crosswalk DataFrame and then merge it with the rest of the data.

    a. To create a crosswalk, define a pandas DataFrame which has five columns: `type` and `subtype` from the original dataset, and three new columns `updated_type`, `updated_subtype`, and `updated_subsubtype`.
    b. Let each row of this DataFrame be a unique combination of `type` and `subtype`. Then, based on the hierarchy you proposed in Q3, fill in `updated_type`, `updated_subtype`, and `updated_subsubtype` accordingly. Remember to name the `NA` subtypes as "Unclassified". Hint: your crosswalk should have 32 observations.

c. Merge the crosswalk with the original data using `type` and `subtype`. How many rows are there for Accident - Unclassified?

d. **EXTRA CREDIT/OPTIONAL**: After merging the crosswalk,can you check that the crosswalk and the new merged dataset have the same values in `type` and `subtype`?

App #1: Top Location by Alert Type Dashboard (30 points)

We will first make a spatial dashboard that displays the top 10 locations in Chicago with the highest number of alerts of a chosen type and subtype. Follow the lecture notes on how to create a Basic Shiny app and create it in a new folder called `top_alerts_map` (**Note: remember to choose "No" when prompted to choose if you would like to use Shiny Express**). Remember to use reactive decorators (e.g., `@reactive_calc`) to avoid unnecessary recalculations.

1. Let's begin by by developing our output outside of Shiny. We will first clean and collapse the data.

   a. The `geo` variable holds coordinates data, but they are stored in a string that represents the Well-Known Text representation of the point. Create two variables `latitude` and `longitude` after extracting the latitude and longitude from the string. *Hint: you will have to use regular expressions or* `regex` *to extract your text. You can look at the tutorial on regex here (here (link)) or prompt ChatGPT to put together a regular expression that extracts the coordinates. If you use ChatGPT, copy your prompt ChatGPT's response below.*

   b. Bin the latitude and longitude variables into bins of step size 0.01. That is, coordinats with values of `(-41.9232, -87.4251)` should become `(-41.92, -87.43)`. Which binned latitude-longitude combination has the greatest number of observations in the overall dataset?

   c. Collapse the data down to the level of aggregation needed to plot the top 10 latitude-longitude bins with the highest number of alerts for a chosen *type and subtype* (Note: no sub-subtype). Save DataFrame as `top_alerts_map.csv` file in the `top_alerts_map` folder you created. What is the level of aggregation in this case? How many rows does this DataFrame have?

2. Using `altair`, plot a scatter plot where the x-axis is latitude and y-axis is longitude, and the points represent the latitude-longitude bins with the 10 highest number of "Jam - Heavy Traffic" alerts. Encode the size of the mark to represent the number of alerts. Hint: for a better presentation of the plot, you should set the domain of the x and y axis to be between some minimum and maximum values for the latitude and longitude.

3. Next, we will layer the scatter plot on top of a map of Chicago.

a. Download the neighborhood boundaries as a GeoJSON from the Chicago Data Portal (link). **EXTRA CREDIT**: can you download the file directly with Python using the `requests` package?

b. Load it into Python using the `json` package and prepare it for Altair using the following code:

```
# MODIFY ACCORDINGLY
file_path = "./top_alerts_map/chicago-boundaries.geojson"
#----

with open(file_path) as f:
    chicago_geojson = json.load(f)

geo_data = alt.Data(values=chicago_geojson["features"])
```

For now on, follow the Altair documentation Altair geographic plots documentation (link) to plot `geo_data` in Altair. **NOTE:** Be particularly careful with your choice of the projection (`project` method) of the map. If you are having trouble with the map showing up correctly, you can try to use the `equirectangular` projection.

4. Layer the scatter plot from step 2 on top of a plot of the map using the information you loaded in step 3 and `geo_data`. Adjust the x and y axis domains so that the two layer correctly on top of each other. You may need to change the layering order of the map and the scatter plot or make the map fill transparent in order to properly see both plots.

5. Now, we are ready to make our data and plot into the Shiny dashboard. In particular, we're going to make a dashboard that lets users select in a single dropdown menu which combination of type and subtype they want to display. Once the user has made their selection, the app will show the 10 locations with the highest counts of those alerts.

    a. For the UI component, create a **single dropdown menu** for type and subtype. Insert a screenshot of the dropdown menu below. How many total type x subtype combinations are there in your dropdown menu?

    b. Recreate the "Jam - Heavy Traffic" plot from above by using the dropdown menu and insert a screenshot of the graph below.

    c. Use your dashboard to answer the following question: where are alerts for road closures due to events most common? Insert a screenshot as your answer below.

    d. Other than the examples above, give an example of a question this dashboard could be used to answer. Formulate the question, take a screenshot of the selection and resulting plot in the dashboard, and then provide the answer.

    e. Can you suggest adding another column to the dashboard to enhance our analysis?

App #2: Top Location by Alert Type and Hour Dashboard (20 points)

1. We will now create a new App folder called `top_alerts_map_byhour`. This new app will modify your first app to add a **slider** to `topalerts_map` that lets users **pick an hour of the day**, and show the top 10 locations at that time of day. But, again, we will first work on the data outside of Shiny before we make the app.

    a. Take a look at the whole dataset we are working with. Given the information present in the `ts` column, would you think that it would be a good idea to collapse the dataset by this column? Why or why not?

    b. Create a new variable called `hour` that extracts the hour from the `ts` column (i.e. if the timestamp is 2024-01-01 **01**:34:32, the `hour` column should be 01:00). Then, generate a new collapsed dataset that has the required columns for us to plot the top 10 locations by hour, type and subtype (i.e. we want to add a new level of aggregation). How many rows does this dataset have? Beware that this might take some time to run. Save this collapsed dataset as `top_alerts_map_byhour.csv` in the `top_alerts_map_byhour` folder.

    Beware: this might take some time to run, but shouldn't take more than 5 minutes

    c. Generate an individual plot of the top 10 locations by hour for 'Jam - Heavy Traffic' for three different times within a day. Don't forget to use the map layer you created while working for the first app and use the same longitude and latitude ranges.

2. We will now turn into creating the Shiny app. As mentioned, for this app we will have a single dropdown menu (similar to the one from App 1) and add a slider to pick the hour. Remember to not use the whole dataset for this app, but the collapsed dataset you created in the previous part.

    a. Create the UI for the app, which should have the dropdown menu to choose a combination of type and subtype, and a slider to pick the hour. Insert a screenshot of the UI below.

    b. Recreate the "Jam - Heavy Traffic" plot from above by using the dropdown menu and slider and insert a screenshot of each plot below.

    c. Use your dashboard to answer the following question: does it seem like road construction is done more during morning hours or night hours? No need to insert more than two screenshots of the dashboard to support your answer.

App #3: Top Location by Alert Type and Hour Dashboard (20 points)

1. As choosing a single hour might not the best way to look at this data, we will now create a new app that builds upon App 2. For this app, we will add a component that allows the user to pick a **range of hours**. For this new app, create a new folder called `top_alerts_map_byhour_sliderrange`. We will modify the app from the previous part to allow the user to go from a slider to a *slider range* – that is, it will allow the user to pick a *range* of hours like 6AM-10AM, rather than a single hour.

    a. Think about what we did in App 1 and 2 regarding collapsing our dataset to make it easier for the Shiny app to handle the data. Given our goal of plotting the top 10 locations by alert type and range of hours, would it be a good idea to collapse the dataset by range of hours? Why or why not?

    b. Before going into the Shiny app, create a plot of the top 10 locations by alert type and range of hours for Jam - Heavy Traffic between 6AM and 9AM.

2. We will now create our new Shiny app adding the slider for the range of hours.

    a. Create the required UI for the App, which should have the dropdown menu to choose a combination of type and subtype, and a slider to pick the hour **range**. Insert a screenshot of the UI below and the plot.

    b. Recreate the "Jam - Heavy Traffic" plot from above by using the dropdown menu and slider range. Insert a screenshot of your App below

3. We will now add a conditional panel to the app to allow the user to toggle between the choice between a slide for a single hour or a slider for a range of hours. For this, we will use a switch button component.

    a. Read the documentation on switch buttons and then add the switch button with the label "Toggle to switch to range of hours" to the app. Insert a screenshot of your App with the addition of the switch button (it doesn't need to be functional yet) and answer the following question: what are the possible *values* (understood as the possible values for `input.switch_button` if the switch button is named `switch_button`) for this switch button?

    b. Modify the UI to add a conditional panel that shows a slider for a single hour when the switch button is toggled. Insert two screenshots of your App with the addition of the conditional panel, demonstrating that when the switch button is toggled, the slider for a single hour is shown and when it is not toggled, the slider for a range of hours is shown.

c. Lastly, modify the UI and server logic to add the functionality to the App so that when the switch button is toggled, the plot we show is the corresponding one according to our choice between hours (single hour or range of hours). Insert two screenshots showing this functionality: a plot generated with the slider for a single hour and a plot generated with the slider for a range of hours using the conditional panel functionality.

d. **EXTRA CREDIT**: No need to code this part. What kind of changes would you make to the app in order for you to achieve a plot similar to the one below?