CTA Service Cuts Problem Set 6

Peter Ganong and Felix Farb

Due Sat Feb 21 at 5:00PM Central "This submission is my work alone and complies with the 30538 integrity policy." Add your initials to indicate your agreement: **___** For an additional 5% bonus (in aggregate, not for each section), we ask you to please answer the following question for each section.
We would like to hear how you used AI (if at all) on this section. Where did you find it useful/not useful? We are asking for knowledge-sharing purposes; your answer will not impact your grade on the assignment.

The data files for PS6 are available here. Download the data files to your repository to complete the problem set. Add the data folder to `.gitignore` so the large data files are not pushed to the repository.

Github Classroom Assignment Setup and Submission Instructions

1. **Accepting and Setting up the PS6 Assignment Repository**
    - Each student must individually accept the repository for the problem set from Github Classroom ("ps6") – https://classroom.github.com/a/NKjugJbC
    - Contents of ps6 assignment repository:
        - `ps6_template.qmd`: this is the Quarto file with the template for the problem set. You will write your answers to the problem set here.
2. **Submission Process**:
    - Knit your completed solution `ps6.qmd` as a pdf `ps6.pdf`.
        - Your submission does not need runnable code. Instead, you will tell us either what code you ran or what output you got.
    - To submit, push `ps6.qmd` and `ps6.pdf` to your ps5 assignment repository. Confirm on Github.com that your work was successfully pushed.
    - Add the data folder to `.gitignore` so large datasets are not pushed to your assignment repository

Grading

- You will be graded on what was last pushed to your PS6 assignment repository before the assignment deadline
- Problem sets will be graded for completion as: {missing (0%); - (incomplete, 50%); (complete, 100%)} You need to push both your `ps6.qmd` and `ps6.pdf` to your repository. Submissions that do not include both files will automatically receive 50% credit.

As of September 2025, the Chicago Transit Authority was facing a major budget shortfall which was going to require a 40% cut in service. Here is the Chicago Sun-Times coverage of the issue. Acting President Nora Leerhsen has asked you to teach her about different ways to achieve that service reduction.[1]

She has told you that she cares about both about efficiency, by which she means affecting the smallest number of riders possible, and equity. Although equity has many different definitions, she suggests that using income is sufficient. You will use the data you cleaned and prepared in PS5.

```
# setup
import geopandas as gpd
from os.path import join
import altair as alt
import pandas as pd
import numpy as np
import time

import warnings
warnings.filterwarnings('ignore')
alt.renderers.enable("png")
alt.data_transformers.disable_max_rows()

data_path = 'data/external'

# improve graph Resolution
import tempfile
from IPython.display import SVG, display, Image
import vl_convert as vlc

def display_altair_png(chart, scale=2):
    """
```

[1] Of course, the same dataset and questions could also be used to think about expanding service in a different budget environment. In this case, by the end of September, after we wrote the problem set, Illinois lawmakers passed a package which covered the CTA funding gap. Details here.

```
    Render an Altair chart to a PNG and display it inline.

    Parameters
    ----------
    chart : altair.Chart
        Altair chart object to render.
    scale : int, optional
        Resolution scaling factor for the PNG (default = 2). Use scale=2 for
↪   standard slides. Use scale = 3-4 for dense figures or PDF exports.
    """
    png_bytes = vlc.vegalite_to_png(chart.to_dict(), scale=scale)

    # Write to a temporary PNG file and display
    with tempfile.NamedTemporaryFile(suffix=".png") as tmp:
        tmp.write(png_bytes)
        tmp.flush()
        display(Image(filename=tmp.name))
```

The following code cells contain your solutions from PS5 (Problems 1-3) that are needed for PS6 to run properly. Paste your PS5 code into these cells.

```
# PS5 Problem 1a: calculate trip counts per route from GTFS data
# Paste your calculate_trip_counts() function and call it here
```

```
# PS5 Problem 1b: read passengers, calculate utilization
# Paste your read_passengers_df(), calculate_passenger_counts(),
# calculate_utilization_rates() functions here, along with the
# line_string_rename mapping and function calls
```

```
# PS5 Problem 3a: spatial join to get rider income
# Paste your rider_income_calc() function here, call it,
# and compute income_quintile
```

```
# PS5 Problem 3b: utilization rates by income quintile
# Paste your calculate_passenger_counts_quintile() and
# calculate_utilization_rates_quintile() functions here, and call them
```

PS6 – equity-efficiency tradeoff and dashboards

Note that submission of PS6 will entail submitted a pdf, qmd, as well as runnable app.py file for the streamlit. We prefer that you use your answers to problem 5 as inputs to problem 6. However, even in the event that you are unable to complete problem 5, we have provided the correct final output files from problem 5 to develop your dashboard in problem 6 in the `data/dashboard` folder.

Problem 5

Next, you will create a menu of service cuts for the CTA chief. Each option should come with information about efficiency and equity consequences.

We will make the following simplifying assumptions:

- Service reductions happen across an entire route without making distinctions of the time of day or the day of the week.
- Every trip has the same income distribution as the route as a whole.

For example, suppose that a route has 4 scheduled trips, 40 passengers total, and 8 passengers from each income quintile. If we cut one trip, then 10 passengers will be affected, 2 from each income quintile.

The crucial assumption we will make is that if a trip is cut, then passengers take another train or bus on that route. Once you have cut a trip, then utilization changes on that route because the affected riders are redistributed to other trips. We have given you a function `cut_trips()` which addresses the redistribution issue without any comments at the end of this script.

Part a. understand `cut_trips()`

To make things manageable, let's start by focusing on just the routes in a minimum working example. Below are the statistics for three bus routes.

	route_id	median_income	passenger_count	utilization_rate
0	11	103611	175	0.433168
1	68	117583	7997	17.890380
2	57	33516	58	0.111538

Read this function carefully and be sure you understand it. Run it on the `mwe` with the `verbose` parameter set to True to see how the function works in this simple example.

Provide two descriptions of the algorithm:

1. Describe it as you would if you had to explain it to someone who is not a master's student in policy
2. Describe it in pseudocode

Part b. record equity impacts

We will consider three measures of passenger welfare

1. *'Passengers affected:'* The number of passengers on routes with longer wait time (because there has been at least some service cut). This metric is equal to zero if there are no cuts to a route and equal to the total number of passengers using the route if there are cuts to a route.
2. *'Average wait time:'* The number of minutes in a day divided by the number of trips along that route in a day, which is averaged up weighted by passenger count in order to get average wait times.
3. *'Passengers with lost service:'* The number of passengers on routes with zero trips.

Name the data frame produced by `cut_trips()` as `df_with_cuts`. Write a new function which takes the data frame `df_with_cuts` and returns two data frames.

1. `df_with_cuts` plus new columns with the overall number of affected passengers, average wait times, and the number of passengers with no service, as well as those same metrics broken down by income quintile.
2. `summary_df` which has the total number of affected passengers and the number of passengers with lost service by income quintile.

Run `compute_cut_impacts(cut_trips(mwe))` to confirm the calculations on your minimal working example.

Part c. propose efficiency-focused cut

Using `cut_trips()`, propose a `new_trip_count` for every route which cuts the lowest-utilization routes first until 40% of trips have been cut.

Here is a skeleton of functions to write:

```
def efficiency(updated_df):

    cut_trips_df = compute_cut_impacts(updated_df)

    return cut_trips_df
```

Part d. propose equity-focused cut

Make a list of trips to cut according to a 'Rawlsian' rule. Rawls proposed maximizing welfare as the well-being of the worst-off citizen. In the context of this problem, we will define that as the well-being of the lowest income quintile. Sort routes by the number of passengers in the route who are in the bottom quintile[2], and cut trips in that order until 40% of trips have been cut.

If you are finding this exercise to be too difficult, you can go to office hours to see on paper a description of pseudo-code that will help with this problem.

Here is a skeleton of functions to write:

```
def cut_trips_low_q1(utilization_quintile_df):

    return updated_df

updated_df_lowq1 = cut_trips_low_q1(utilization_quintile_df)

def efficiency_low_q1(updated_df_lowq1):

    cut_trips_df = compute_cut_impacts(updated_df_lowq1)

    return cut_trips_df
```

Part e. propose balancing equity and efficiency

In this part, you will create a menu of cuts which prioritizes a balance of equity and efficiency. The idea of trading off efficiency and equity considerations should be familiar to you from micro I in the core, when you looked at maximizing different social welfare functions and explored how different social welfare functions would lead to different policy allocations. In this problem we will choose one specific way to trade off efficiency and equity which is that we will create a 'score'.

The score is utilization rate divided by median rider income.

Order routes according to this score, and cut trips from routes with the lowest score until 40% of trips have been cut. Explain why cutting routes with the lowest score is implementing an equity-efficiency tradeoff.

Here is a skeleton of functions to write:

[2]One might also think to order routes by the portion of passengers on the route who are in the bottom quintile.

```
def cut_trips_equity_minded(utilization_quintile_df):
    return updated_df

updated_df_equityminded = cut_trips_equity_minded(utilization_quintile_df)

def efficiency_equity_minded(updated_df_equityminded):

    cut_trips_df = compute_cut_impacts(updated_df_equityminded)

    return cut_trips_df
```

If you are finding this exercise to be too difficult, you can go to office hours to see on paper a description of pseudo-code that will help with this problem.

Part f. summarize impacts of each approach

Coding

1. Using the full sample with all the routes, make a table comparing the number of passengers affected by income quintile under each method. What do you notice?
2. Make the same table two more times, but using metrics of number of passengers with no service, as well as average wait times.

Diagnostic: first run your code on the minimum working example. Below are the results which you should expect for that example.

		Passengers affected by quintile				
Method	Total passengers affected	Q1	Q2	Q3	Q4	Q5
40% cut by lowest utilization	100	40	10	6	9	36
40% cut by lowest Q1 passengers	191	29	11	20	12	120
40% cut by equity minded	71	43	10	1	8	9

		Passengers without service by quintile				
Method	Total passengers without service	Q1	Q2	Q3	Q4	Q5
40% cut by lowest utilization	0	0	0	0	0	0
40% cut by lowest Q1 passengers	175	17	8	20	10	120
40% cut by equity minded	58	42	9	0	7	0

Method	Average wait time (Δ vs baseline)	Average wait time by quintile (Δ vs baseline)				
		Q1	Q2	Q3	Q4	Q5
Baseline	22.58	22.44	22.54	22.62	22.55	22.61
40% cut by lowest utilization	23.44	27.04	23.66	22.94	23.20	22.88
	(0.86)	(4.60)	(1.12)	(0.32)	(0.65)	(0.27)
40% cut by lowest Q1 passengers	22.42	22.61	22.52	22.33	22.51	22.36
	(-0.16)	(0.17)	(-0.02)	(-0.30)	(-0.04)	(-0.25)
40% cut by equity minded	22.59	22.26	22.51	22.68	22.54	22.66
	(0.01)	(-0.18)	(-0.03)	(0.06)	(-0.01)	(0.05)

Thinking

3. What are the advantages of the three different metrics in assessing the policies? Discuss the limitations of the assumptions that we have made, both stated and unstated.
4. Using the tables, analyze the difference between the policies.
5. Discuss the effectiveness of the 'Rawlsian' rule.

In considering how to answer these questions, here is a small concrete example of how to read the MWE tables:

Looking at the equity-minded row in Table 1, notice that it affects only 71 total passengers — fewer than the 100 under efficiency. At first glance, this seems like the equity-minded policy is strictly better. But Table 2 reveals a cost: 58 passengers lose service entirely under equity-minded, compared to zero under efficiency. The equity-minded policy concentrates cuts on fewer routes (eliminating some outright) rather than spreading them, which lowers the total affected count but creates a worse outcome for riders on the eliminated routes.

Use this kind of cross-table reasoning when analyzing the full sample results.

Problem 6 streamlit

To summarize your proposals, you will now create a dashboard using the Streamlit package to display key information in an organized and interactive way.

Part a.

First, visualize each of your three proposals.

1. Using the data from `trips_post_cut_efficiency.csv`, plot new utilization rates against median income using your plotting function from Problem 4. More specifically, modify the plotting function to color routes in one of three colors depending on whether their service is unchanged, reduced, or cut entirely.
2. Repeat this plot again for part 5d using `trips_post_cut_equityminded.csv`
3. Repeat this plot again for part 5e using `trips_post_cut_lowq1.csv`

Similar to Problem 4, your function should be in this form:

```
def plot_utilization_vs_income(
    df: pd.DataFrame,
    title_name: str,
    income_col: str = "median_income",
    util_col: str = "utilization_rate",
    filename: str = "utilization_plot.html",
    color_by_change: bool = False,
):

    return chart
```

Part b.

These static visualizations are helpful, but the major advantage of dashboards are in the interactivity they provide to the user, which allows for a more compact and effectively communicated display of information.

Make a dashboard in streamlit where the user can choose between different policy proposals, whether to show routes with no cuts at all, and whether to show bus, L, or both.

Three components:

1. (easiest) render the plot you made in part (a)
2. (easy) render a table where each row is a route and you provide other helpful columns of information
3. a map that displays the CTA Bus and L lines. Use `pydeck` with the included GEOJSON files for Bus and L routes. Be sure to join the routes data with the trips data correctly. Color each line colored by the number of cut trips on each route, with line thickness determined by the number of affected passengers on each line (using data from the `cut_trips_detail` csv files).

For both the plot and the map, there should be a tooltip giving relevant information on highlighted points/routes.

Here is a skeleton of functions to write:

```
DASHBOARD_DIR = "data/dashboard"
EXTERNAL_DIR = "data/external/dashboard"

TRIPS_FILES = {
    "Efficiency": f"{DASHBOARD_DIR}/cut_trips_detail.csv",
    "Equity-minded": f"{DASHBOARD_DIR}/cut_trips_detail_equityminded.csv",
    "Rawlsian": f"{DASHBOARD_DIR}/cut_trips_detail_lowq1.csv",
}

FIG_FILES = {
    "Efficiency": f"{DASHBOARD_DIR}/trips_post_cut_efficiency.csv",
    "Equity-minded": f"{DASHBOARD_DIR}/trips_post_cut_equityminded.csv",
    "Rawlsian": f"{DASHBOARD_DIR}/trips_post_cut_lowq1.csv",
}

RAIL_GEOJSON = f"{EXTERNAL_DIR}/CTA_-_'L'_(Rail)_Lines_20250924.geojson"
BUS_GEOJSON = f"{EXTERNAL_DIR}/CTA_-_Bus_Routes_20250924.geojson"
```