CTA Service Cuts Problem Set 5 and 6

Peter Ganong and Felix Farb

PS5 and PS6 are much more challenging than PS1, PS2, PS3, and PS4. It is much closer to the type of work you will do for the final project.

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. Here are the datasets you will use:

- Scheduled trips from Chicago Transit Authority in General Transit Feed Specification (GTFS)
- Passengers' actual trips on CTA in Chicago from MIT Transit Lab. These were recently used in the economics paper "Optimal Urban Transportation Policy: Evidence from Chicago' '. We are grateful to one of the authors (Juan Camilo Castillo) for sharing it with us. The appendix describes the data in a bit more detail.
- Median Household income by Census tract from the 2023 5 year ACS. Taken from the NHGIS website.

Tips:

- Read everything (PS5 and PS6) before starting. That will help you to see the big picture before you dive in.
- We encourage you to work mainly in `.py` scripts and then port everything to a `.qmd` submission at the end. This is also the recommended workflow for the final project.

[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.

PS5 – data cleaning, spatial and visualizing the equity-efficiency tradeoff

Due Sat Feb 14 at 5:00PM Central

The data files for PS5 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.

Tips for PS5:

- Problems 1 and 2 are data cleaning and merging. They can be done before next week's lecture. Problem 3 requires spatial data tools. You can also take a pass at problem 4 before next week's lecture as well by generating a random number for each route's average income. Then, after you have finished problem 3, go back and do it with the correct income for each route.
- We expect that your time allocation will be about one-third on problems 1 and 2, one-half on problem 3, and one-sixth on problem 4.

Github Classroom Assignment Setup and Submission Instructions

1. **Accepting and Setting up the PS5 Assignment Repository**
    - Each student must individually accept the repository for the problem set from Github Classroom ("ps5") – https://classroom.github.com/a/J5umMK45
    - Contents of ps5 assignment repository:
        - `ps5_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 `ps5.qmd` as a pdf `ps5.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 `ps5.qmd` and `ps5.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 PS5 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 `ps5.qmd` and `ps5.pdf` to your repository. Submissions that do not include both files will automatically receive 50% credit.

Problem 1 – how well-utilized is each route?

Part a – number of scheduled vehicle trips

Calculate the number of scheduled bus and train trips each week. Collapsing `trips.txt` to the route level and attach labels using `routes.txt`. Do this work with a function `calculate_trip_counts()` which returns a dataframe called `routes_w_trips`.

Required diagnostics to check that you are on track:

- Print the number of routes with schedules in `routes.txt` and the number of routes with actual trips in `trips.txt`.
- Count if there are any routes with trips that don't have schedules and vice-versa. If a route does have no trips, then fill in the number of trips as being zero.
- Expect 133 rows in the combined dataframe.

Part b – number of passenger trips

Next, calculate the number of passenger trips along each route using `bus_passengers.csv` and `train_passengers.csv`.

Required diagnostics to check that you are on track:

- When first reading each dataset, print the number of rows. Each row corresponds to one boarding. Many journeys require multiple boardings.
- Which dataset has journeys that mix different modes (e.g. a train boarding followed by a bus boarding) within a single journey?
- Combine the two datasets. You should have 1236508 rows in the combined dataframe of bus and train passenger trips.

Then, calculate each route's utilization rate (passenger count divided by scheduled trip count).[2] Requirements:

- You will need to find a suitable variable to construct the number of passengers on each route.
- Merge diagnostics: print which scheduled routes are missing in the passenger routes data, and which passenger routes are missing in the scheduled routes data.
- Examine the list you printed carefully. Are there any examples where the same route has different names in the schedule data and the passenger data? If needed, harmonize the names.
- Produce matched data frame `utilization_df`. You should see 126 matched routes.

Here is a skeleton of functions to write:

```
# Function that makes a dataframe of bus/train passengers.
def read_passengers_df()
    # ...
    return passengers_df

# Function to count the number of passengers who take each route. Takes in a
 ↪  combined dataframe of bus and train passengers
def calculate_passenger_counts(passengers_df):
    # ...
    return passenger_count_df

def calculate_utilization_rates(passengers_df):

    passenger_count_df = calculate_passenger_counts(passengers_df)
    # ...

    return utilization_df
```

Problem 2 – which routes have the lowest utilization? (efficiency)

First, calculate the 40th percentile of trips by utilization rate. For this calculation, assume each trip has the same utilization rate as the route as a whole.

[2] One slightly awkward thing: the utilization rates are *relative*. What this means is that we do not know (and indeed Juan Camilo who gave us the data said he did not know) how many days of data contribute to this data file. This is the kind of thing that happens very frequently with data—you save just the columns that you need for your analysis and columns that a future user might want are lost. Fortunately, even without knowing the dates used for the data and therefore not knowing the absolute utilization rate, we still are able to figure out which routes are more or less in use, which is the information the CTA chief needs to decide which routes should see cuts in service.

Second, make a histogram of trips (not routes) with 30 bins of the utilization rate. Suggestions and questions:

- Feel free to use pandas or altair manipulation functions – whatever you find easier.
- You will notice some trips with far higher utilization rates that will mess with the histogram if you don't winsorize the data. What do those trips correspond to?
- Taking the 40th percentile of trip-weighted utilization as a cutoff, color the bars below the 40th percentile cutoff in red. Given the binning structure, it is impossible to color exactly 40% of the trips red. If you use utilization bins of width 1, you will find that you can color either 37.9% or 42.5% of the trips red.

What are the 5 routes with lowest utilization rates?

Here is a skeleton of functions to write:

```
# Function to return the 40th percentile of trips by utilization
def weighted_quantile(values, weights, q):

    return p40_utilization


def make_hist(utilization_df):

    return figure
```

Problem 3 – income of riders on each route (equity)

Next, we want to get information on the income distribution of CTA riders and how it varies by route. The passenger data do not have information on income (they are anonymized), but they do have information on where they got on. Assume that passenger income equals median income in the Census tract where they boarded.[3]

This problem has a few things that make it tricky.

- We have put the data from Census in the problem set repo, but you will need to figure out which variable to use.
- Use the income at the location where the trip begins. In a multi-boarding trip, this may be different from the Census tract in which they boarded the vehicle.

[3]Obviously, this is a heroic assumption. It helps make the problem set manageable.

Part a.

1. Make the passenger data into a spatial data frame using the provided longitude and latitude columns.
2. Spatially join the passenger observations to census tracts, and use this to assign an income to each passenger.
3. Assign each passenger into one of five equally-sized groups by tract-level income. These five groups are quintiles of the passenger income distribution. Add these quintiles as a column to `passengers_inc_df`. You will use this calculation when you do problem 5 next week.

Details:

- Compute the number of unique longitude and latitude points. You should find that there are 353 points.
- Do the spatial join using each *unique* point instead of every observation. This cuts down the number of points to match from 558,894 to just a few hundred points which therefore speeds up computation.
- 7 points have negative reported income. Be sure to drop the points. Can you figure out why those locations have negative reported income? Google could be helpful here.
- Because the longitudes and latitudes are in degrees instead of meters, we should set them to a *geographic* coordinate reference system (CRS) like `EPSG:4326`.
- Print the CRS of the gdf with the census tracts, and compare it to the CRS that you chose for the passenger data. Then transform the CRS of one to the other before performing the spatial join.

Here is a skeleton of functions to write:

```
def rider_income_calc(passengers_df):
    tracts_path = os.path.join(in_path,
↪   "illinois_tract_income/illinois_tract_income.shp")
    tracts = gpd.read_file(tracts_path)

    return passengers_inc_df
```

Part b.

Collapse `passengers_inc_df` to compute the number of passengers from each income quintile on each route. Then repeat the utilization rate calculation from problem 1. Note that now you will have six utilization rates – one for each income quintile and an overall rate.

Required diagnostics to check that you are on track:

- Add a test in your code to verify that the number of passengers from each income quintile sums to the total number of passengers with non-missing income.
- The set of passengers with non-missing income is smaller than the set of all passengers which you computed in problem 1 for two reasons. How much smaller and why?
- How many route-quintile cells do not have any passengers (e.g. the 76 bus does not have any low-income riders)? Be sure to fill these in as zero.
- Do you think it makes more sense to store the data as a wide format (one row per route with columns for each quintile) or a long format (one row per route-quintile)? Make a guess now and feel free to revisit as you keep working on the problem set.
- Sanity check your results: which route has the most bottom income quintile passengers? Which route has the most top income quintile passengers?

Here is a skeleton of functions to write:

```
def calculate_passenger_counts_quintile(passengers_inc_df):

    return passenger_count_df

def calculate_utilization_rates_quintile(passengers_inc_df):
     passenger_count_df =
 ↪  calculate_passenger_counts_quintile(passengers_inc_df)
    trips_count_df = calculate_trip_counts()

    return utilization_quintile_df
```

Problem 4

Take the two variables you have lovingly constructed in the last two problems and help Nora understand how they are related for each route. For this problem you will start from a data frame with utilization (efficiency, problem 2) and median rider income (equity) for each route.

1. Compute a route's income as the median of the income of all the riders who use that route (as estimated by tract of origin). This is slightly different from and simpler than what you did in problem 3[4]
2. Make a scatter plot with route utilization on the y-axis and the median rider income on the x-axis. Find three routes which you think your readers might be particularly interested in knowing the income and utilization levels for before deciding whether to cut service. Annotate them using red and label them with the `route_long_name` column we got from `routes.txt`.

[4]Your work on problem 3 will come in handy next week as part of problem 5!

3. Draw a horizontal line at the 40th percentile of trip-weighted utilization (which you calculated already in problem 2). This shows which routes would be cut if you prioritized only efficiency and ignored equity.
4. The plot is hard to read because the low utilization routes are all tightly compressed. Make a second version of the plot where the y-axis is on a reverse log scale (we didn't cover this in class, but it is easy to find via the online altair documents). Compare the two versions of the plot. For which audiences would you want to use the regular y-axis and for which audiences would you want to use the reverse-log y-axis?

Here is a skeleton of functions to write:

```
def plot_utilization_vs_income(
    df: pd.DataFrame,
    log_scale: bool,
    income_col: str = "median_income",
    util_col: str = "utilization_rate",
    add_cutoff: bool = True,
    filename: str = "utilization_plot.html",
    highlight_ids=("1", "2", "3"),
):
```

PS6 – equity-efficiency tradeoff and dashboards

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_lowq1):

    cut_trips_df = compute_cut_impacts(updated_df_lowq1)

    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[5], 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:

[5] 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_equalize_quintiles(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	3.23	3.21	3.22	3.23	3.22	3.23
40% cut by lowest utilization	3.34	3.84	3.37	3.28	3.31	3.27
	(0.11)	(0.63)	(0.15)	(0.05)	(0.09)	(0.04)
40% cut by lowest Q1 passengers	3.48	3.49	3.35	3.57	3.33	3.52
	(0.25)	(0.28)	(0.13)	(0.34)	(0.11)	(0.29)
40% cut by equity minded	3.32	3.81	3.36	3.24	3.30	3.24
	(0.09)	(0.60)	(0.14)	(0.01)	(0.08)	(0.01)

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 similar analysis for equity minded policy in the minimum working example:

The equity minded policy has the lowest number of passengers affected, even when compared with the lowest utilization policy. This is because the score used to determine which trips to cut does not change when more trips are cut on a given route, which agrees with the linear cost to cutting trips on a route that is assumed by the passengers affected metric.

When looking at the other two metrics, equity minded appears to cuts the most from the lowest quintiles. This is because utilization rate and income are highly correlated in this small subsample. As a result, the Laramie route is cut heavily despite its low income, while Lincoln and particularly Northwest Highway are cut less despite their higher incomes, due to their utilization rates both being substantially higher.

```
corr = mwe["median_income"].corr(mwe["utilization_rate"])
print("Correlation between income and utilization rate:", corr)
```

```
Correlation between income and utilization rate: 0.6404743594290565
```

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 with a map that displays the CTA Bus and L lines, 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).The dashboard should also give the plot using the function written in part (a).

For both the plot and the map, there should be a tooltip giving relevant information on highlighted points/routes. The user should be able to select which policy proposal is displayed on the dashboard.

For the maps, use pydeck with the included GEOJSON files for Bus and L routes. Be sure to join the routes data with the trips data correctly.

Here is a skeleton of functions to write:

```
TRIPS_FILES = {
    "Efficiency maximizing": "cut_trips_detail.csv",
    "Equalize quintiles": "cut_trips_detail_equal.csv",
    "Rawlsian": "cut_trips_detail_lowq1.csv",
}

FIG_FILES = {
    "Efficiency maximizing": "trips_post_cut_efficiency.csv",
    "Equalize quintiles": "trips_post_cut_equal.csv",
    "Rawlsian": "trips_post_cut_lowq1.csv",
}


INPUT_DIR = "issues/issue_12_pset_first_pass/input"
RAIL_GEOJSON = f"{INPUT_DIR}/CTA_-_'L'_(Rail)_Lines_20250924.geojson"
BUS_GEOJSON = f"{INPUT_DIR}/CTA_-_Bus_Routes_20250924.geojson"
```