

Calculating scheduled bus volumes from the TfL API using Python

Cambridge Spark Bootcamp Project
William Petty



Why calculate scheduled bus volumes?



- Transport for London (TfL) operates one of the world's most extensive public bus networks, with 675 routes, a fleet of c. 9,300 vehicles, and 19,000+ bus stops
- Londoners rely extensively on buses for surface public transport – buses account for 1 journey in 8, with a daily ridership of 5 million
- However, due to their size and speed, buses also present a risk to vulnerable road users
- Particularly in the case of planning for cycling, knowing the volume of buses expected to use a particular part of the road network can inform decisions about the most appropriate interventions to keep road users safe

<https://tfl.gov.uk/corporate/about-tfl/what-we-do>

<https://data.london.gov.uk/dataset/tfl-bus-stop-locations-and-routes>

<https://tfl.gov.uk/info-for/media/press-releases/2023/november/latest-tfl-figures-show-the-tube-reaching-4-million-journeys-per-day>

What public data is available?

Gets the timetable for a specified station on the give line

[GET](#) /Line/{id}/Timetable/{fromStopPointId}

Parameters

fromStopPointId
string
The originating station's stop point id (station naption code e.g. 940GZLUASL, you can use /StopPoint/Search/{query} endpoint to find a stop point id from a station name)

id
string A single line id e.g. victoria

Test this endpoint

[TRY](#)

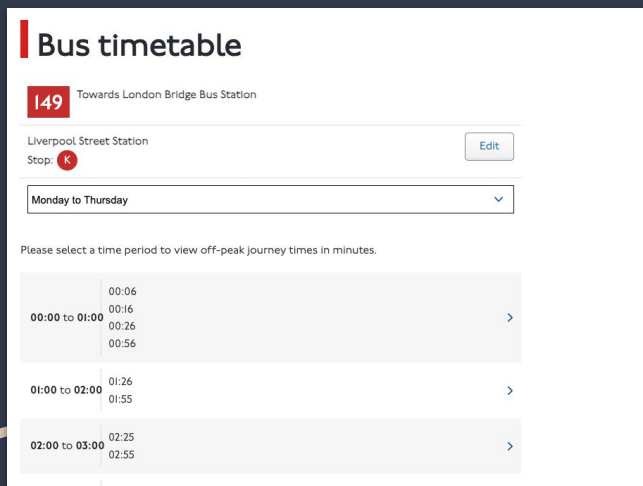
Response Type

RESPONSE SAMPLE

```
{
  "lineId": "string",
  "lineName": "string",
  "direction": "string",
  "pdUrl": "string",
  "stations": [
    {
      "routeId": 0,
      "parentId": "string",
      "stationId": "string",
      "id": "string",
      "topMostParentId": "string",
      "direction": "string",
      "towards": "string",
      "nodes": [
        "string"
      ],
      "stopType": "string",
      "stopLetter": "string",
      "zone": "string",
      "accessibilitySummary": "string",
      "hasDisruption": true,
      "lines": [
        {
          "id": "string",
          "name": "string",
          "url": "string",
          "fullName": "string",
          "type": "string",
          "crowding": {
            "passengerFlows": [
              {
                "timeSlice": "string",
                "value": 0
              }
            ]
          }
        }
      ]
    }
  ]
}
```

- Highway authorities carry out traffic counts using automated measuring devices such as pneumatic tubes, and use the data to build a picture of the characteristics of road sections
- However, in the case of buses, TfL makes daily timetables available, which could avoid the need for observations in most cases
- Timetable data could make bus volume calculations accessible to organisations such as campaign groups
- Bus timetable information can be accessed through TfL's Unified API:
<https://api.tfl.gov.uk/>
- Due to its intended use by passengers, the information in the API is organised around finding a given bus time, rather than total volumes of bus traffic

The aims of this project



The screenshot shows a web application titled "Bus timetable". It features a red bus icon with the number "149" and the text "Towards London Bridge Bus Station". Below this, the "Stop" is listed as "Liverpool Street Station" with a red circular icon containing a white "K". An "Edit" button is located to the right of the stop name. A dropdown menu is set to "Monday to Thursday". A message states: "Please select a time period to view off-peak journey times in minutes." Below this, there are three time period rows, each with a range of times and a right-pointing arrow:

Time Period	Journey Times (minutes)
00:00 to 01:00	00:06, 00:16, 00:26, 00:56
01:00 to 02:00	01:26, 01:55
02:00 to 03:00	02:25, 02:55

- I will write Python code that accesses the TfL API, and returns timetable information about bus timetables for a given bus stop or group of bus stops
- The information will be presented as a graph, showing the number of buses per hour – for each route – stopping at the given bus stop or group of bus stops as per the timetable
- As a default the graph will show an entire week's timetable, but users can select a smaller range of days if required
- Functionality will be included to input an optional title for the graph; otherwise the title will be the list of bus stop IDs

Some notes on how the code works

```
# The main function that pulls the data from TfL's API and graphs it
def get_bus(stop_id_list, title = 'default', start_day = 1, end_day = 7):

    stop_dict = make_stop_dict(stop_id_list)

    # create an empty list to hold the data rows
    all_data = []

    # create the API timetable URL for each bus for each stop in the dictionary
    for stop_id in stop_dict:
        for bus in stop_dict[stop_id]:
            URL = "https://api.tfl.gov.uk/Line/" + bus + "/Timetable/" + stop_id
            R = requests.get(URL)
            J = json.loads(R.text)

            try:
                for route in J["timetable"]["routes"]:
                    for schedule in route["schedules"]:

                        for journey in schedule["knownJourneys"]:
                            journeyrow = []
                            journeyrow.append(stop_id)
                            journeyrow.append(bus)
                            journeyrow.append(schedule["name"])
                            journeyrow.append(journey["hour"]#.zfill(2))
                            journeyrow.append(journey["minute"]#.zfill(2))
                            all_data.append(journeyrow)

            # A KeyError occurs when a stop-bus pair is the last stop on the line - no 'timetable' key
            except KeyError:
                print(f'No timetable found for bus {bus} at stop {stop_id}')

    df = pd.DataFrame(columns=['stop', 'bus', 'dayName', 'hour', 'minute'], data=all_data)
```

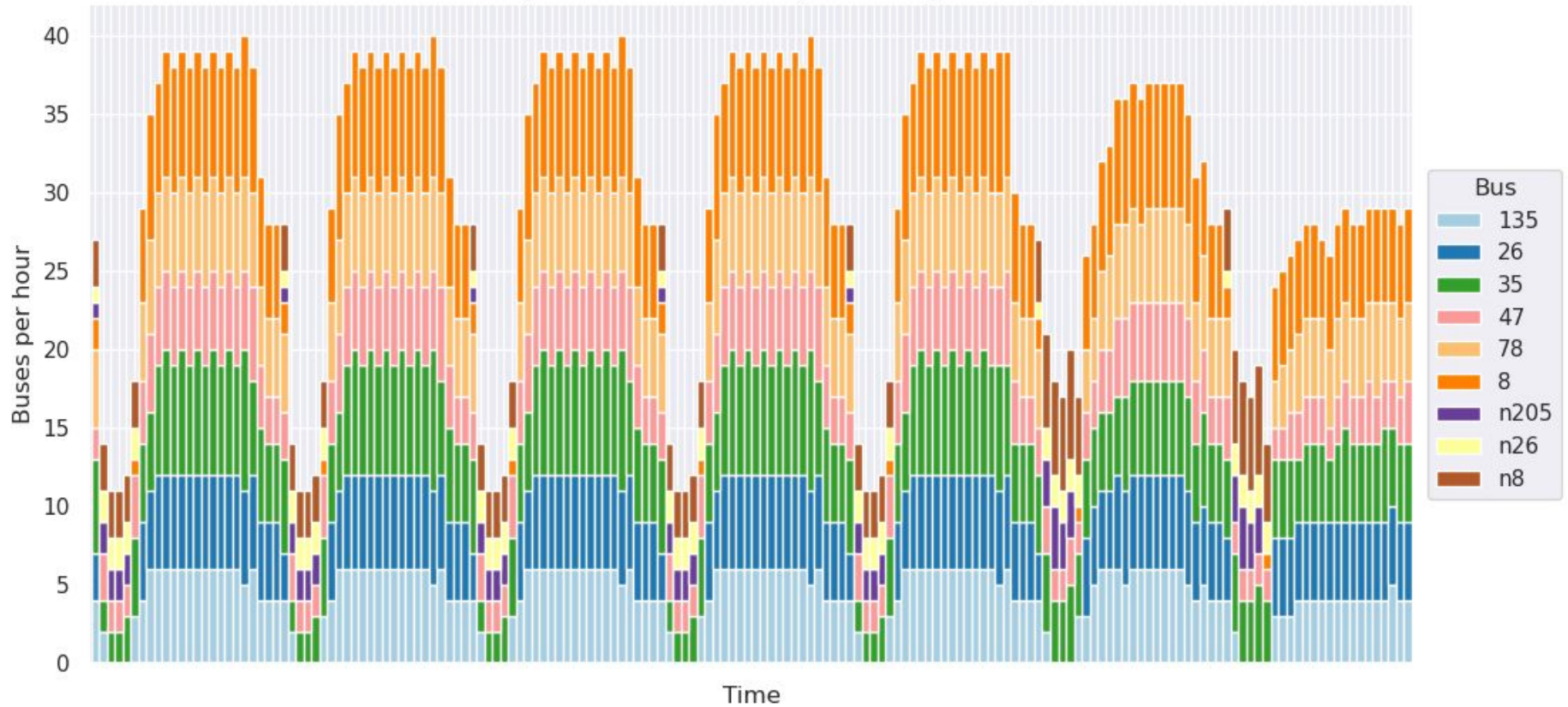
- The API identifies bus stops using a unique ID called a NapTAN code; the Python code takes input in the form of a list of NapTAN codes
- Using this input, the Python code accesses the API, and compiles a list of bus routes that stop at each bus stop
- Then it makes a second API call to get a set of timetables for each route/bus stop combination
- The timetables for each bus stop on the list are combined; henceforth all stops on the list are treated as a single entity
- Days with identical timetables are listed as one entry in the API (e.g. Monday to Thursday), so any timetables that cover multiple days need to be duplicated for each day
- Finally the number of buses that stop each hour of the week can be counted and graphed

Results

```
# An example with multiple bus stops and selected days of the week  
get_bus(['490000138F', '490000138E', '490000138L', '490000138K'], start_day=1, end_day=5, \  
        title='Liverpool Street Station, both directions')
```

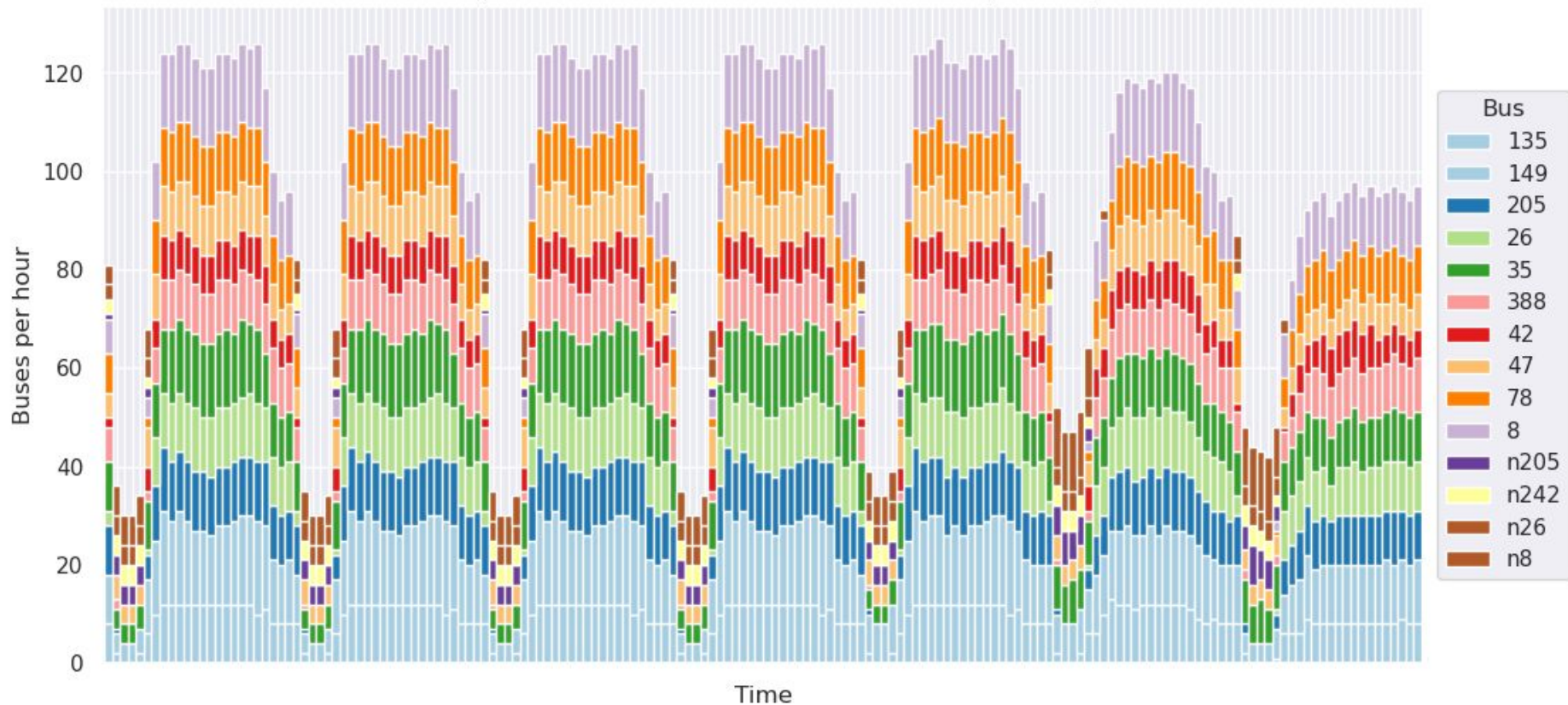


Liverpool Street Station, Stop F, Monday-Sunday



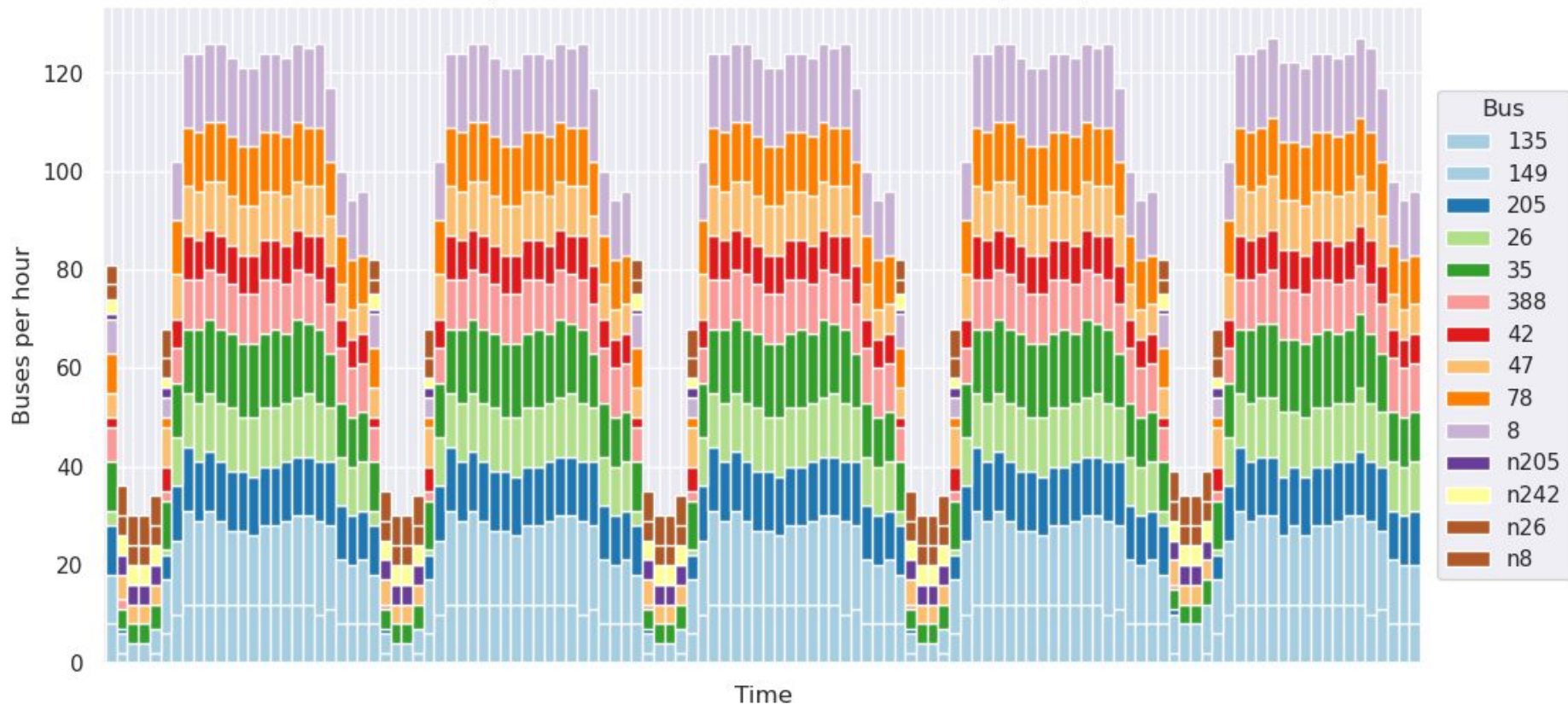
An example with one bus stop and all days of the week

Liverpool Street Station, both directions, Monday-Sunday



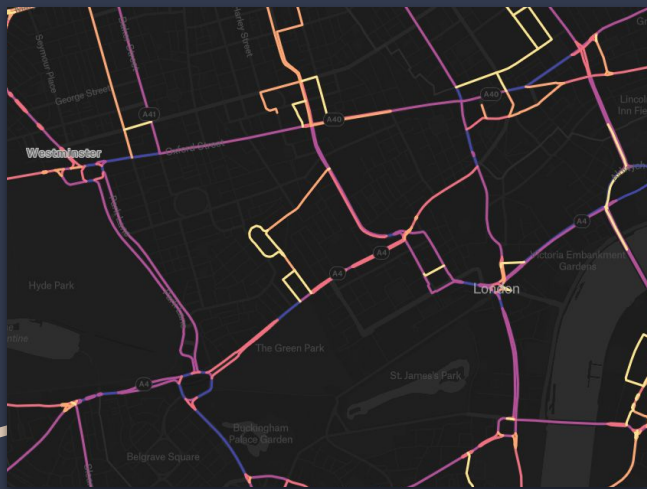
An example with multiple bus stops and all days of the week

Liverpool Street Station, both directions, Monday-Friday



An example with multiple bus stops and selected days of the week

Limitations and next steps



- The need to know the NapTAN code for each bus stop is currently a limit to useability. Selecting from a list of common names (e.g. Liverpool St Stop F) or selection from a map would be an improvement
- Because of the purpose of the API, the final stop on a bus's route has no timetable (as no passengers will get on). This means that if one of the stops is a bus's terminus, the total will be an undercount
- The code could be modified to get the number of weekly services for every London bus route (instead of per hour and per bus stop). Combining this with bus route GIS data from OpenStreetMap, it would be possible to create a map of bus volumes (either weekly or as a daily average) on every London road with a bus route