# Students in Motion: A Visual Analysis of MRT and Bus Services to Singapore Polytechnic

Oh Tien Cheng (2012072)
DAAA/1B/01

# Objectives

Analyse the differences in peak passenger volume to SP between MRT & Buses

Determine if it is necessary to increase bus frequencies for buses going to SP

# Utility Functions

```python
def plotAllOutliers(df: DataFrame, cols: Union[List[str], None] = None) -> Figure:
    cols = df.select_dtypes(include=np.number).columns.tolist() if cols is None else cols
    fig, ax = plt.subplots(len(cols), 1, figsize = (10, 6), tight_layout = True)
    for idx, col in enumerate(cols):
        sns.boxplot(df[col], ax = ax[idx])
    return fig
```

Plot All Outliers as Boxplots

```python
def outlierDetection(df : DataFrame, var : str, sort : bool = True) -> DataFrame:
    """
    Utilty function for detecting basic outliers. Prints out no. of outliers, shows first 5 outliers and displays a
box plot. Outliers are defined in this case by Tukey's fences, where outliers are data points above or below the
upper or lower quantiles by 1.5 * IQR.

    Parameters:
        df = DataFrame to detect outliers in
        var = name of column where outlier detection is performed
        sort = should the returned dataset be sorted?
    Returns:
        Dataset which only includes outliers
    """
    Q3 = df[var].quantile(0.75)
    Q1 = df[var].quantile(0.25)
    IQR = Q3 - Q1
    UpperFence = Q3 + 1.5 * IQR
    LowerFence = Q1 - 1.5 * IQR
    mask = (df[var] > UpperFence) | (df[var] < LowerFence)
    outlier_df = df[mask]
    if len(outlier_df) == 0:
        print("No Outliers")
    else:
        print("Outliers in Series (First 5)")
        print(outlier_df.head())
        print("No. of Outliers:", len(outlier_df))
    sns.boxplot(y = var,data = df, orient= "h")
    sns.despine(left = True)
    plt.title(f"Box Plot of {var}", fontdict = {
        "weight" : "semibold"
    })
    plt.show()
    if sort:
        return outlier_df.sort_values(var)
    else:
        return outlier_df
```

Outlier Detection (Returns a Dataframe containing ONLY the outliers for further analysis)

```python
def removeOutliers(data: Union[DataFrame, Series], cols : Union[List[str], None] = None) -> DataFrame:
    df_type = type(data)
    assert df_type is DataFrame or df_type is Series, "data should either be a pandas dataframe or series"
    assert cols is None or type(cols) is list, "Either provide None, or a list of col names"
    if cols is not None:
        data = data[cols]
    Q3 = data.quantile(0.75)
    Q1 = data.quantile(0.25)
    IQR = Q3 - Q1
    print("Shape Before Removing Outliers:", data.shape)
    data = data[~((data < (Q1 - (1.5 * IQR))) | (data > (Q3 + (1.5 * IQR)))).any(axis=1)]
    print("Shape After Removing Outliers:", data.shape)
    return data
```

Remove Outliers

**How** have the commuting habits of Singaporeans changed over the years?

# Dataset 1: Public Transport Utilisation – Average Daily Public Transport Ridership

- Average daily number of trips made islandwide on MRT, LRT, bus & taxi.
- Data covers January 1, 1995 to December 31, 2016
- 88 Rows and 3 Columns
- 10 Missing Values
- Only data from 2001 onwards had been updated when the methodology of estimating taxi ridership was revised in 2003.

```
Data columns (total 3 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   year                    88 non-null     int64
 1   type_of_public_transport  88 non-null     object
 2   average_ridership       78 non-null     float64
```

| | year | type_of_public_transport | average_ridership |
|---|---|---|---|
| 0 | 1995 | MRT | 740000 |
| 1 | 1995 | LRT | 0 |
| 2 | 1995 | Bus | 3009000 |
| 3 | 1995 | Taxi | 0 |
| 4 | 1996 | MRT | 850000 |

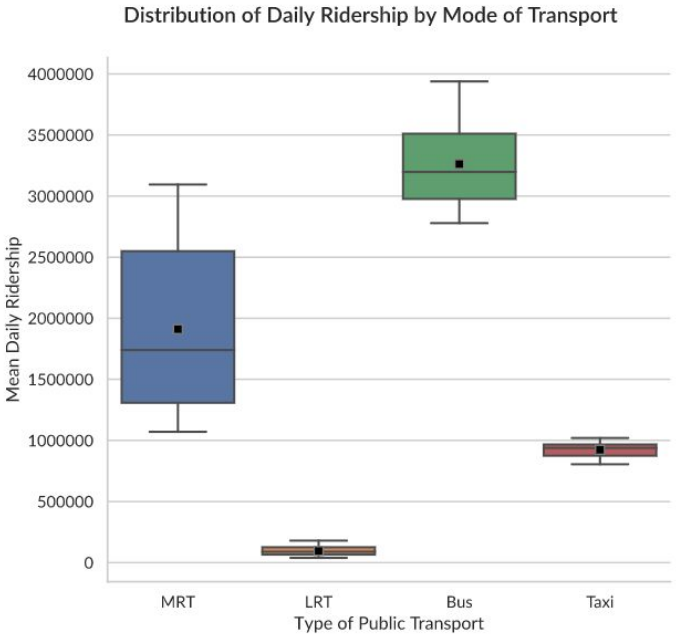
# Data Preprocessing

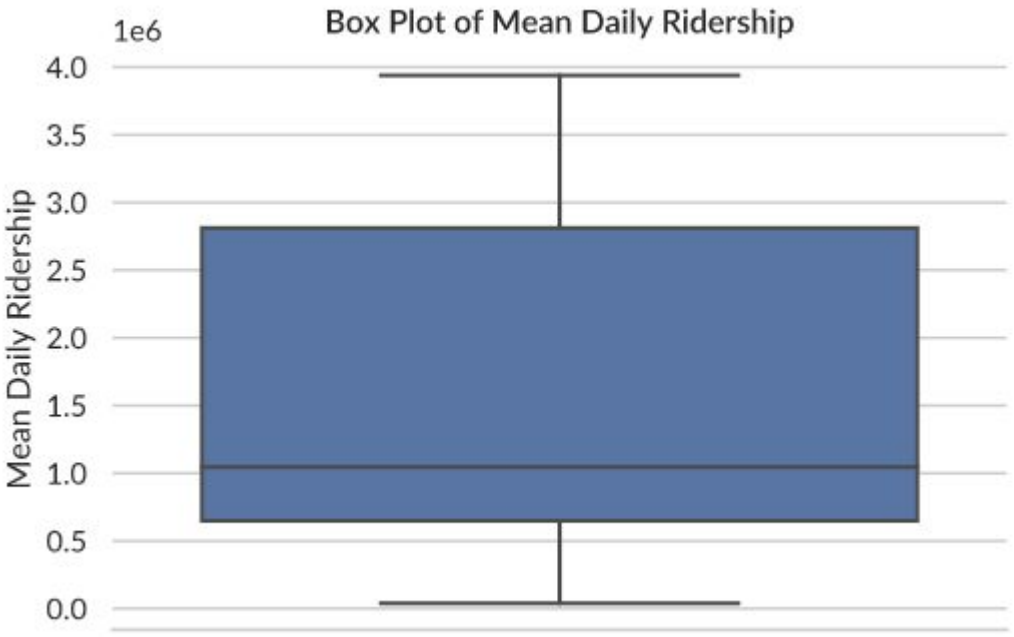
1. Data loaded using pd.read_csv() with na_values = 0 to treat all 0's as NA values
2. Missing data checked for using .isna().sum()
3. All years before 2001 dropped using boolean indexing
4. Dataframe columns renamed using .rename()
5. outlierDetection function used to check "Mean Daily Ridership" (overall) for outliers
6. sns.catplot used to generate box plots of ridership by mode of transit for outliers.
7. Mean Daily Ridership converted to millions based scale

Graph 1:

- Data grouped by year and summed using .groupby and .sum

# Checking for Outliers

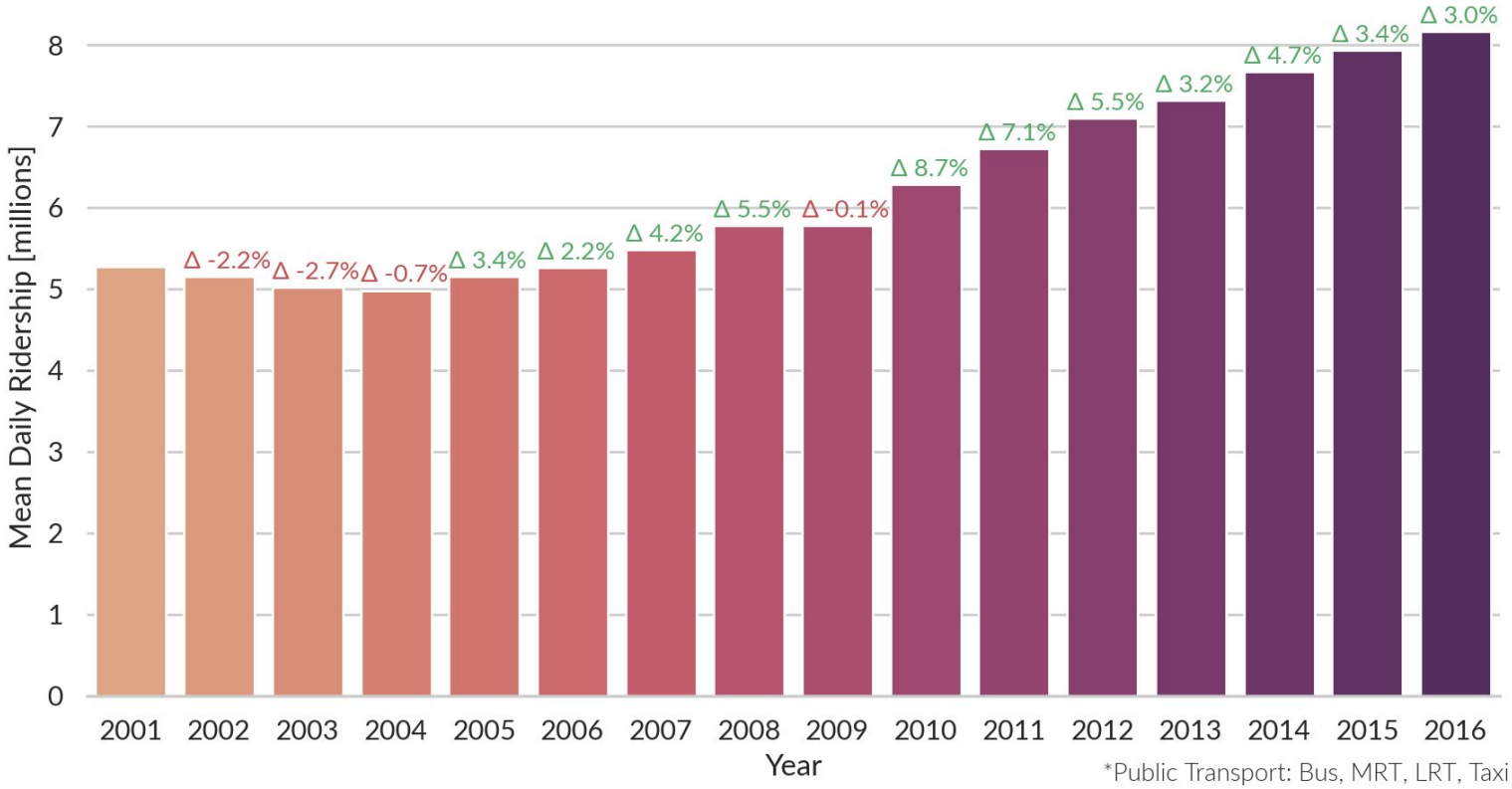



No outliers could be found in Dataset 1

```
# Data Preprocessing
daily_ridership["Mean Daily Ridership [millions]"] = daily_ridership["Mean Daily Ridership"] / 1000000 # Create a
new column using a millions based scale for ridership
total_ridership = daily_ridership.groupby("Year").sum() # get total ridership for each year
total_pct_change = total_ridership["Mean Daily Ridership"].pct_change() * 100 # calculate percentage change over the
previous year

fig, ax = plt.subplots(figsize = (9, 5), tight_layout = True)

# Plot
sns.barplot(x = total_ridership.index, y= "Mean Daily Ridership [millions]", data = total_ridership, palette=
"flare", ax = ax)

# Annotation & Visuals
ax.set_title("Mean Daily Public Transport Ridership in Millions", loc = "left", fontsize = 16, weight="semibold")
ax.text(11, -1.05, "*Public Transport: Bus, MRT, LRT, Taxi", weight="light", fontsize = 10)
sns.despine(left = True)
pct_annot = [ax.text(s = f"\u0394 {pct_change:.1f}%", x = idx - 0.4,  y= y + 0.1, fontsize = 10, color = 'r' if
pct_change < 0 else 'g') for idx, (y, pct_change) in enumerate(zip(total_ridership["Mean Daily Ridership
[millions]"].iloc[1:], total_pct_change.iloc[1:]), 1)]
fig.show()
fig.savefig("./plots/mean_annual_ridership.png", dpi = 200)
```

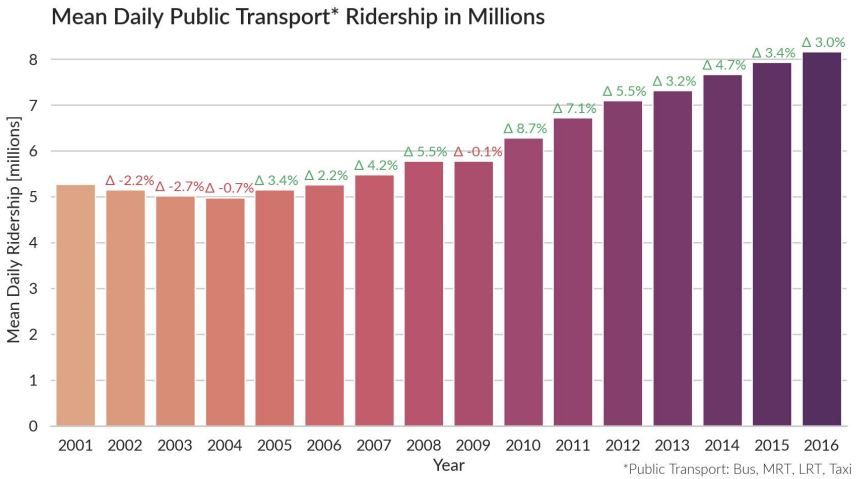
Graph 1 Code




Is public transport becoming more popular?

# Analysis

- Mean daily ridership has increased from **5.2 million** in 2001 to **8.1 million** in 2016.
- Greatest decrease in 2003 by 2.7% due to outbreak of SARS virus.
- Notably, this increase in ridership appears to have outpaced population growth in Singapore. According to SingStat, S'pores population was approx 4.1 million in 2001, growing to 5.6 million in 2016. (inclusive of non-citizens). This is an increase of 1.5 million (**+36.6%**), compared to an increase of 2.9 million in ridership (**+55.8%**).
- Suggests that increase in ridership is not just due to population growth.

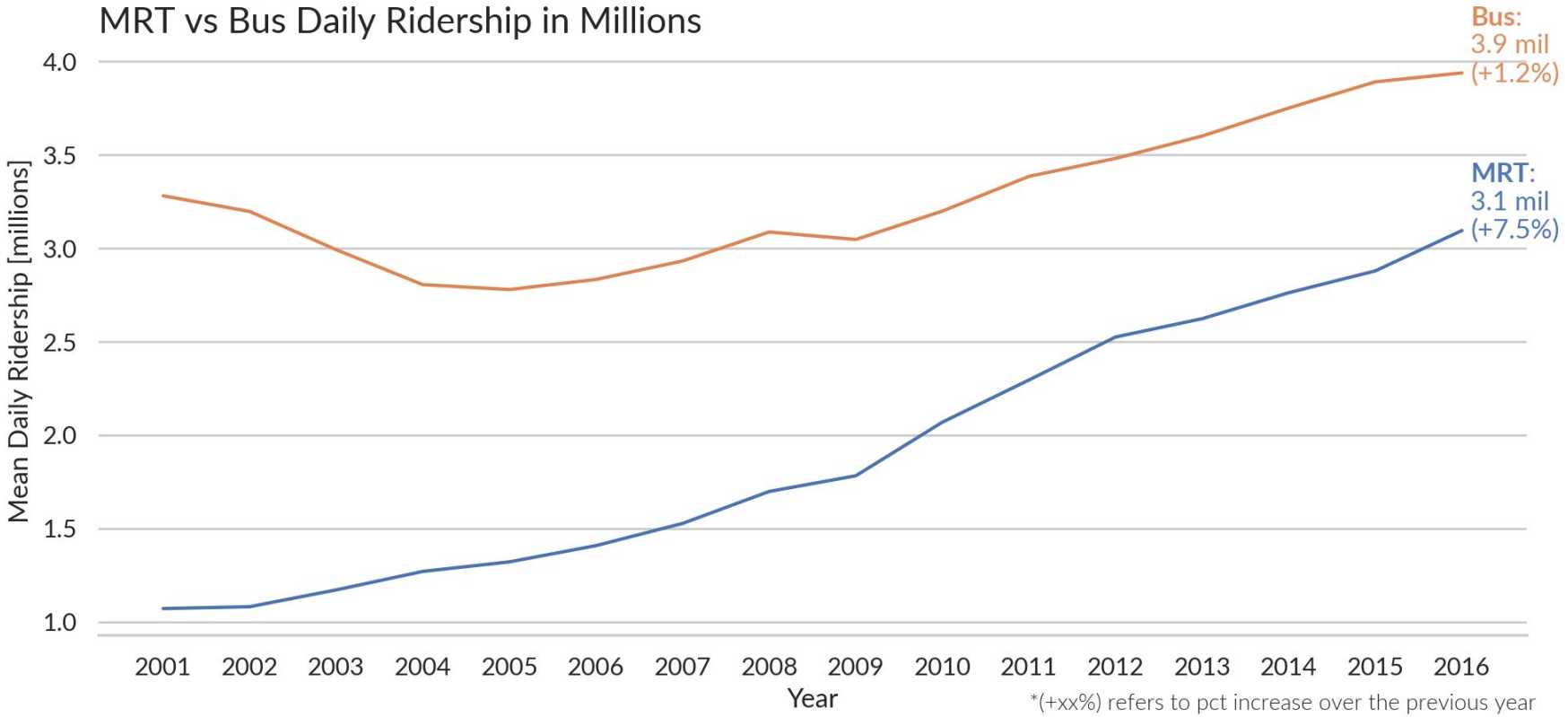


```
# Data Preprocessing
daily_ridership_df_busmrt = daily_ridership[daily_ridership["Type of Public Transport"].isin(["Bus", "MRT"])]
ridership_2016 = daily_ridership_df_busmrt.pivot("Year", "Type of Public Transport", "Mean Daily Ridership
[millions]").iloc[-1, :]
bus2016, mrt2016 = ridership_2016
percentage_change = daily_ridership_df_busmrt.pivot(index = "Year", columns = "Type of Public Transport", values =
"Mean Daily Ridership").pct_change() * 100
pct_change_bus, pct_change_mrt = percentage_change.iloc[-1, :]

# Plotting
fig, ax = plt.subplots(figsize=(10, 5), tight_layout = True)
sns.lineplot(x = "Year", y= "Mean Daily Ridership [millions]", hue = "Type of Public Transport", data =
daily_ridership_df_busmrt,ax = ax, legend = None)


# Annotation & Visuals
ax.annotate(r"$\bf{Bus}$:" + "\n{:.1f} mil\n(+{:.1f}%)".format(bus2016, pct_change_bus), (2016, bus2016), (2016.1,
3.89), color="#df8d5f")
ax.annotate(r"$\bf{MRT}$:" + "\n{:.1f} mil\n(+{:.1f}%)".format(mrt2016, pct_change_mrt), (2016, mrt2016), (2016.1,
3.05), color = "#5378b3")
ax.set_title(r"$\bf{More\ Singaporeans\ are\ Taking\ the\ MRT}$" + "\nMRT vs Bus Daily Ridership in Millions", loc =
"left", fontsize = 16)
ax.text(2011, 0.53, "*(+xx%) refers to pct increase over the previous year", weight="light", fontsize = 10)
ax.grid(False, axis = "x") # remove vertical grid lines
# ax.legend(loc = "lower right", title = "Mode of Transport")
ax.set_xticks(daily_ridership["Year"].unique())
# ax.set_ylim(bottom = 0)
sns.despine(left = True)
fig.show()
```

## Graph 2 Code

## More Singaporeans are Taking the MRT

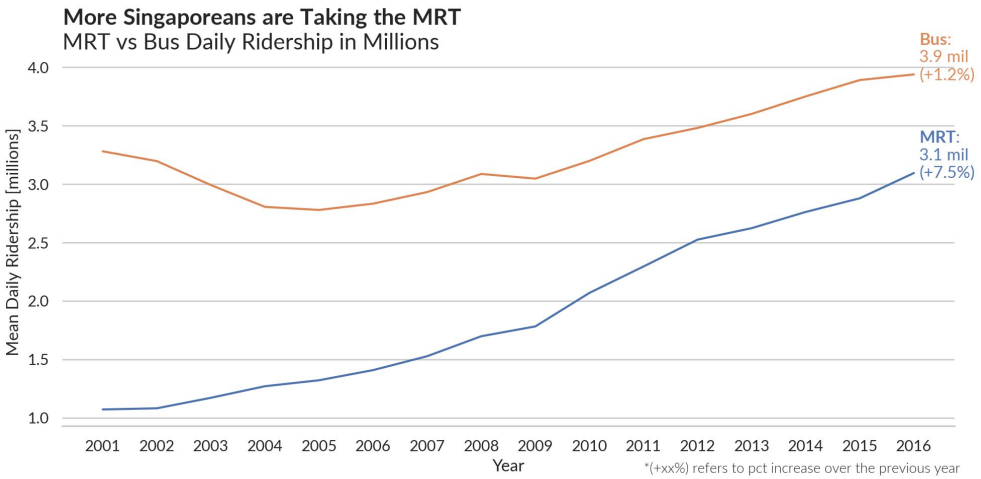
MRT vs Bus Daily Ridership in Millions




MRT is rapidly catching up to buses, but buses are still the most popular option.

# Analysis

- Buses remained the most popular mode of public transit from 2001 to 2016, with a peak ridership of ~3.9 million in 2016.
- Sharp decline in Bus Ridership in 2004, due to SARS outbreak.
- However, the MRT is rapidly rising in popularity. The percentage increase in ridership over the previous year was 7.5% for the MRT in 2016, compared to only 1.2% for buses.
- Could be explained by increased accessibility of the MRT; New MRT lines like the Downtown Line, and upgrades to existing stations.



**When** and **what** is the peak demand for students travelling to SP via MRT?


*Source: Wikimedia Commons*

Students travelling by MRT to SP drop off at Dover MRT Station

# Dataset 2: Passenger Volume By Train Stations

| # | Column | Non-Null Count | Dtype |
|---|---|---|---|
| 0 | YEAR_MONTH | 18984 non-null | object |
| 1 | DAY_TYPE | 18984 non-null | object |
| 2 | TIME_PER_HOUR | 18984 non-null | int64 |
| 3 | PT_TYPE | 18984 non-null | object |
| 4 | PT_CODE | 18984 non-null | object |
| 5 | TOTAL_TAP_IN_VOLUME | 18984 non-null | int64 |
| 6 | TOTAL_TAP_OUT_VOLUME | 18984 non-null | int64 |

- Dataset retrieved from LTA Datamall API
- Tap in and tap out passenger volume by weekdays and weekends for individual train stations
- By 15th of every month, the passenger volume for previous month data will be generated
- *TIME_PER_HOUR*: refers to the hour of the day. E.g. 15 = 1500H to 1559H
- For some train interchanges, the station codes will be merged and considered as one station (E.g. EW14/NS26 refers to Raffles Place station)
- Data collected covers October 2020 to December 2020 (Jan 2021 data unavailable)
- No missing data was found.
- Merged Data has 18984 rows and 7 columns

| | YEAR_MONTH | DAY_TYPE | TIME_PER_HOUR | PT_TYPE | PT_CODE | TOTAL_TAP_IN_VOLUME | TOTAL_TAP_OUT_VOLUME |
|---|---|---|---|---|---|---|---|
| 0 | 2020-10 | WEEKDAY | 11 | TRAIN | NS7 | 2353 | 1912 |
| 1 | 2020-10 | WEEKENDS/HOLIDAY | 11 | TRAIN | NS7 | 1434 | 1940 |
| 2 | 2020-10 | WEEKDAY | 16 | TRAIN | SW4 | 1033 | 1457 |
| 3 | 2020-10 | WEEKENDS/HOLIDAY | 16 | TRAIN | SW4 | 514 | 522 |
| 4 | 2020-10 | WEEKDAY | 10 | TRAIN | CC5 | 1319 | 3515 |

First 5 Rows

| | YEAR_MONTH | DAY_TYPE | TIME_PER_HOUR | PT_TYPE | PT_CODE | TOTAL_TAP_IN_VOLUME | TOTAL_TAP_OUT_VOLUME |
|---|---|---|---|---|---|---|---|
| count | 18984 | 18984 | 18984.000000 | 18984 | 18984 | 18984.000000 | 18984.000000 |
| unique | 3 | 2 | NaN | 1 | 159 | NaN | NaN |
| top | 2020-11 | WEEKDAY | NaN | TRAIN | CG1/DT35 | NaN | NaN |
| freq | 6328 | 9499 | NaN | 18984 | 120 | NaN | NaN |
| mean | NaN | NaN | 13.346660 | NaN | NaN | 8091.070375 | 8055.466761 |
| std | NaN | NaN | 6.100588 | NaN | NaN | 12979.937998 | 13053.073201 |
| min | NaN | NaN | 0.000000 | NaN | NaN | 0.000000 | 0.000000 |
| 25% | NaN | NaN | 9.000000 | NaN | NaN | 942.000000 | 960.000000 |
| 50% | NaN | NaN | 14.000000 | NaN | NaN | 3006.500000 | 2997.000000 |
| 75% | NaN | NaN | 19.000000 | NaN | NaN | 9447.750000 | 9202.500000 |
| max | NaN | NaN | 23.000000 | NaN | NaN | 172514.000000 | 149809.000000 |

Descriptive Statistics

```python
def get_transport_data_zip(url : str, dates: List[str], zipName: str, token: str, skipDownload : bool = False) ->
DataFrame:
    """
    Utility function for getting zip file from LTA Datamall api, extracting data, then merging them together. This
only works on certain LTA datamall apis: those that return a link to download a zip file. It also assumes all files
inside the zip are csv files.

    Parameters:
        url = API url for get request
        dates = List of dates where each date has the format "YYYYMM", tells datamall which dates to get data from
        zipName = Specify the name of the zipfile to be saved. Final name of file will be zipName+date.zip
        token = LTA Datamall API Key
        skipDownload = if file is already downloaded, we can skip downloading again to avoid hitting the API rate
limit

    Returns:
        Merged dataset from all dates
    """
    try:
        assert type(url) is str, "url should be a string"
        assert type(dates) is list, "dates should be a list of dates where each date has the format 'YYYYMM', tells
datamall which dates to get data from"
        assert type(zipName) is str, "Specify the name of the zipfile to be saved. Final name of file will be
zipName+date.zip"
        assert type(token) is str, "Token should be the LTA Datamall API Key as a string"
        assert type(skipDownload) is bool, "skipDownload should be a boolean value indicating if download of zip
file should be skipped"
        files = []
        for date in dates:
            if not skipDownload: # if file not already downloaded, download it from datamall
                req = requests.get(url, params = { "Date" : date }, headers = { "AccountKey" : token })
                print(req)
                req = req.json()
                link = req["value"][0]["Link"]
                req = requests.get(link, allow_redirects = True) # request will redirect to download link
                with open("{}{}.zip".format(zipName, date), "wb") as f: # creating a new file
                    f.write(req.content) # write file sent from request
            with zipfile.ZipFile("{}{}.zip".format(zipName, date)) as myzip:
                files += myzip.namelist() # get a list of all csv files in the zip file
                myzip.extractall() # extract all files
        data = [pd.read_csv("{}".format(filename)) for filename in files]
        return pd.concat(data, ignore_index= True) # return merged data
    except Exception as error:
        print(error)
        print(req)
        raise error
```

# Get Transport Data

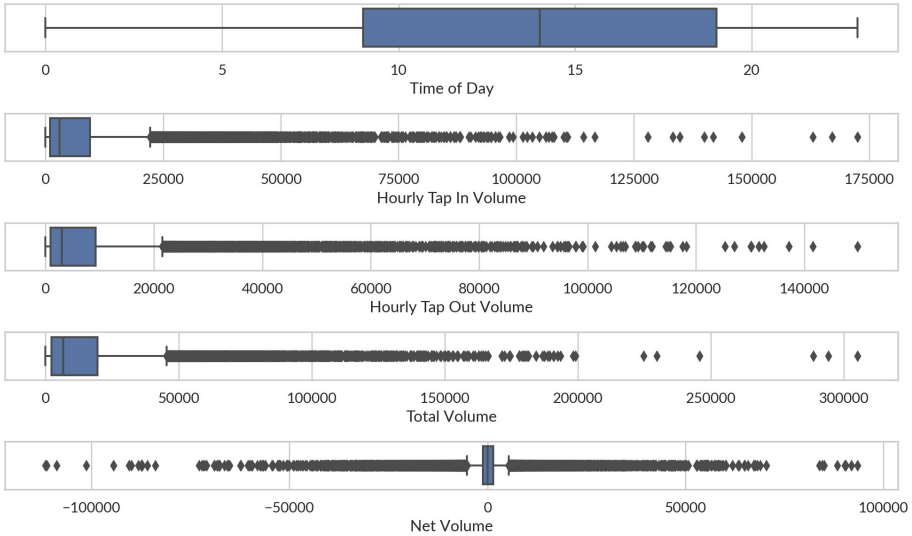
# Data Preprocessing

Retrieving the Data & Processing

1. A custom function, get_transport_data_zip, was used to query the Datamall API (using requests library), download the zip files for each month available, extract them (zipfile library), read the csv (pd.read_csv) and merge the dataframes together (pd.concat)
2. Null values checked for using .isna().sum(), and summary statistics generated using .describe()
3. Dataframe columns renamed using .rename, and Total Volume & Net Volume columns created using Hourly Tap In and Out data
4. Data filtered using boolean indexing and .isin() to include only Dover, Jurong East and Raffles Place MRT
5. Name column added to map station name to code (using .apply)

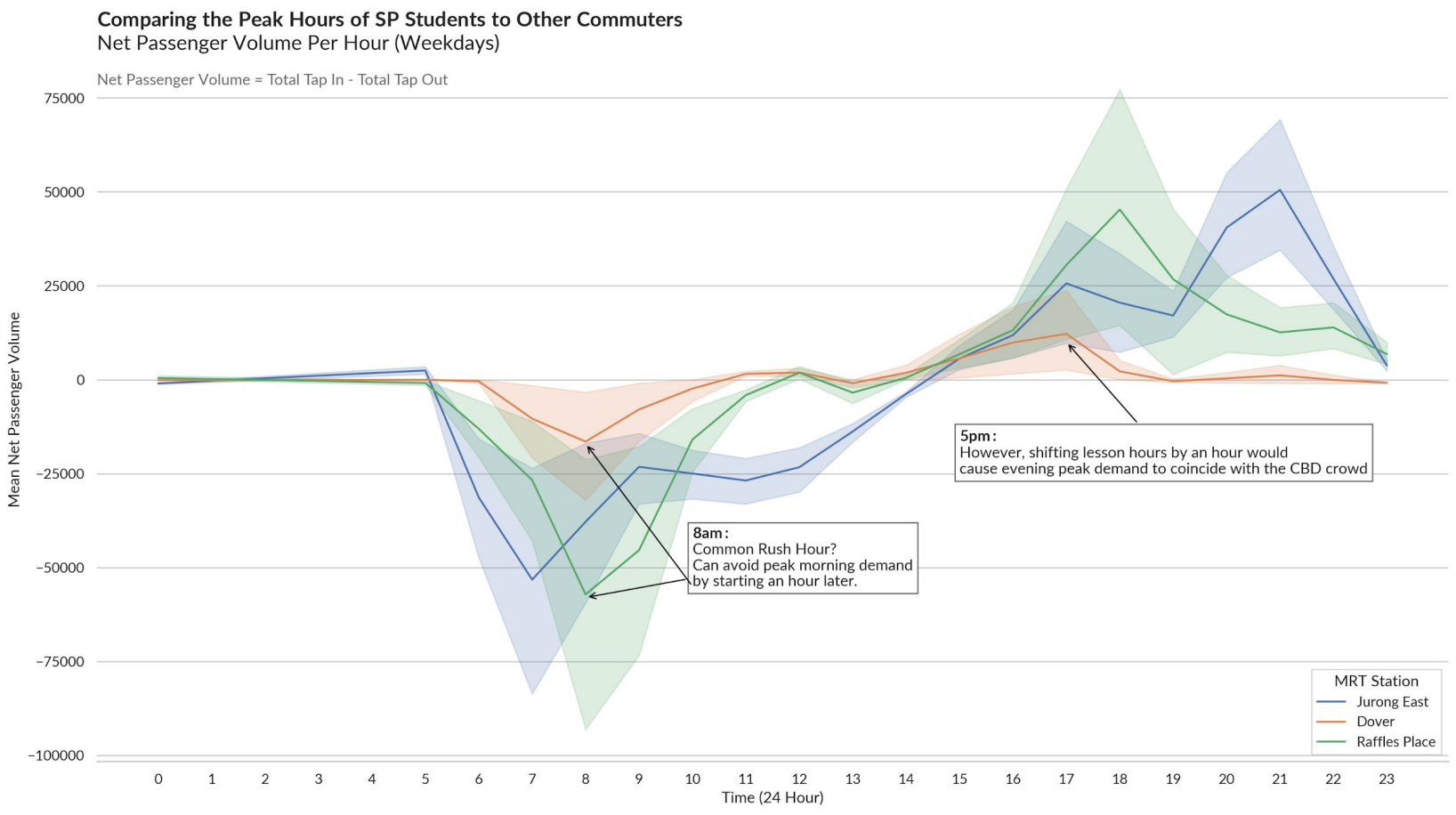
# Outliers

- Many outliers present in the dataset
- There are similar numbers of outliers in both tap in and out volumes. All outliers are those that are above the upper fence.
- Outliers are not a mistake, they represent the fact that certain stations experience extremely high peak demand.
- Many outliers are from MRT interchanges, where two lines meet; Thus having more people than a normal station.
- In this case, as outliers provide useful information, they will not be removed.



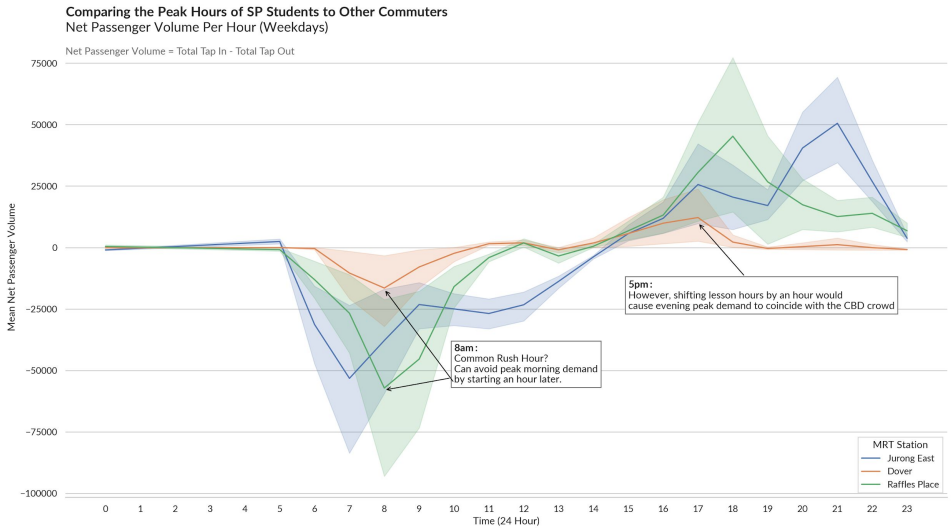
```
fig, ax = plt.subplots(figsize = (16, 9), tight_layout = True)
sns.lineplot("Time of Day", "Net Volume", "Name", data = stations_comparison_df, ax = ax)

ax.set_title(r"$\bf{Between\ a\ Rock\ And\ A\ Hard\ Place}$" + "\nNet Passenger Volume Per Hour (Weekdays)", loc =
"left", fontsize = 16)
ax.grid(False, axis = "x") # remove vertical grid lines
ax.set_xticks(np.arange(24)) # make tick every hour
ax.set_xlabel("Time (24 Hour)")
ax.set_ylabel("Mean Net Passenger Volume")
ax.legend(title = "MRT Station", fancybox = False, loc="lower right")

# Annotations
bbox = {
    "boxstyle" : "square,pad=0.3",
    "fc" : "white",
    "ec" : "dimgrey"
}
ax.text(-1.15, 78500, "Net Passenger Volume = Total Tap In - Total Tap Out", color = "dimgray")
ax.annotate(r"$\bf{8am:}$" + "\nCommon Rush Hour?\nCan avoid peak morning demand\nby starting an hour later.", (8,
-58000), (10, -55000), arrowprops = { "arrowstyle" : "->", "color" : "black"}, bbox = bbox)
ax.annotate("", (8, -17000), (10, -55000), arrowprops = { "arrowstyle" : "->", "color" : "black"})
ax.annotate(r"$\bf{5pm:}$" + "\nHowever, shifting lesson hours by an hour would\ncause evening peak demand to
coincide with the CBD crowd", (17, 10000), (15, -25000), arrowprops = { "arrowstyle" : "->", "color" : "black"},
bbox = bbox)
sns.despine(left = True)
fig.show()
```

Graph 3 Code




When is the peak demand for SP students?

# Analysis




- Net volume follows a “valley and hill” shape, where in the morning, most movement is out of the stations, and in the afternoon/evening, most movement is into the stations (going home).
- Peak net volume for Jurong East (7am) and Raffles Place (8am) is more than 50k each, which is more than twice the peak net volume for Dover MRT (8am).
- Most people arrive at Dover from 8:00am to 8:59am, and leave at 4:00pm to 4:59pm. This likely reflects typical class scheduling with most lessons starting around 8 to 10am.
- The peak demand hours for Dover (8am and 4pm) coincide with Raffles Place (8am) in the morning and are just before Raffles Place (5am) in the evening (reflecting 9-5 working hours). To escape the morning rush hour, would need to start more lessons later, but lessons would have to end at around 7 (to escape evening rush hour for Raffles).

**How** does the passenger volume for buses to SP **compare** to the MRT?

# Dataset 3: Passenger Volume By Bus Stops

```
 #   Column               Non-Null Count   Dtype
---  ------               --------------   -----
 0   YEAR_MONTH           580862 non-null  object
 1   DAY_TYPE             580862 non-null  object
 2   TIME_PER_HOUR        580862 non-null  int64
 3   PT_TYPE              580862 non-null  object
 4   PT_CODE              580862 non-null  int64
 5   TOTAL_TAP_IN_VOLUME  580862 non-null  int64
 6   TOTAL_TAP_OUT_VOLUME 580862 non-null  int64
dtypes: int64(4), object(3)
memory usage: 31.0+ MB
```

- Dataset retrieved from LTA Datamall API
- Tap in and tap out passenger volume by weekdays and weekends for bus stops
- By 15th of every month, the passenger volume for previous month data will be generated
- *TIME_PER_HOUR*: refers to the hour of the day. E.g. 15 = 1500H to 1559H
- Data collected covers October 2020 to December 2020 (Jan 2021 data unavailable)
- No missing data was found.
- 580862 Rows and 7 Columns

| | YEAR_MONTH | DAY_TYPE | TIME_PER_HOUR | PT_TYPE | PT_CODE | TOTAL_TAP_IN_VOLUME | TOTAL_TAP_OUT_VOLUME |
|---|---|---|---|---|---|---|---|
| 0 | 2020-10 | WEEKENDS/HOLIDAY | 17 | BUS | 45379 | 148 | 82 |
| 1 | 2020-10 | WEEKDAY | 17 | BUS | 45379 | 443 | 294 |
| 2 | 2020-10 | WEEKENDS/HOLIDAY | 13 | BUS | 80051 | 1201 | 1144 |
| 3 | 2020-10 | WEEKDAY | 13 | BUS | 80051 | 2382 | 2754 |
| 4 | 2020-10 | WEEKDAY | 13 | BUS | 5319 | 78 | 317 |

First 5 Rows

| | Year-Month | Day Type | Time of Day | Mode of Travel | PT_CODE | Hourly Tap In Volume | Hourly Tap Out Volume |
|---|---|---|---|---|---|---|---|
| count | 580862 | 580862 | 580862.000000 | 580862 | 580862.000000 | 580862.000000 | 580862.000000 |
| unique | 3 | 2 | NaN | 1 | NaN | NaN | NaN |
| top | 2020-12 | WEEKDAY | NaN | BUS | NaN | NaN | NaN |
| freq | 194409 | 293337 | NaN | 580862 | NaN | NaN | NaN |
| mean | NaN | NaN | 13.315412 | NaN | 48777.359948 | 453.011736 | 453.008901 |
| std | NaN | NaN | 6.105520 | NaN | 25466.697828 | 1578.477199 | 1501.728528 |
| min | NaN | NaN | 0.000000 | NaN | 1012.000000 | 0.000000 | 0.000000 |
| 25% | NaN | NaN | 9.000000 | NaN | 25359.000000 | 23.000000 | 25.000000 |
| 50% | NaN | NaN | 14.000000 | NaN | 49159.000000 | 118.000000 | 126.000000 |
| 75% | NaN | NaN | 18.000000 | NaN | 67089.000000 | 407.000000 | 412.000000 |
| max | NaN | NaN | 23.000000 | NaN | 99189.000000 | 122459.000000 | 100153.000000 |

Descriptive Statistics

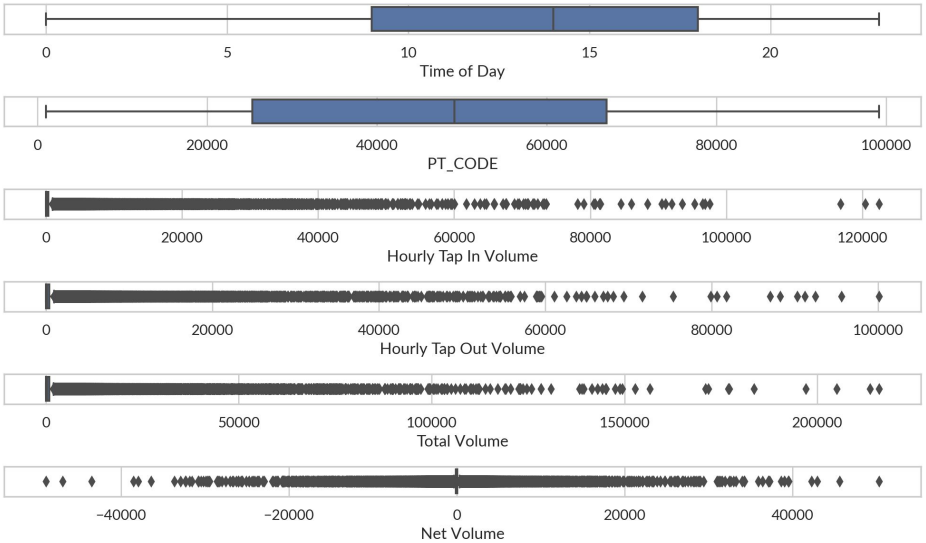
# Data Preprocessing

Retrieving the Data & Processing

1. get_transport_data_zip was used to retrieve the dataset
2. Null values checked for using .isna().sum(), and summary statistics generated using .describe()
3. Dataframe columns renamed using .rename, and Total Volume & Net Volume columns created using Hourly Tap In and Out data
4. Outliers checked for.
5. Name column added to map bus stop name to code (using .apply)
6. pd.Concat is used to merge this dataframe with the MRT volume dataset
7. Data filtered using boolean indexing and .isin() to include only the Dover Bus Stop & Dover MRT

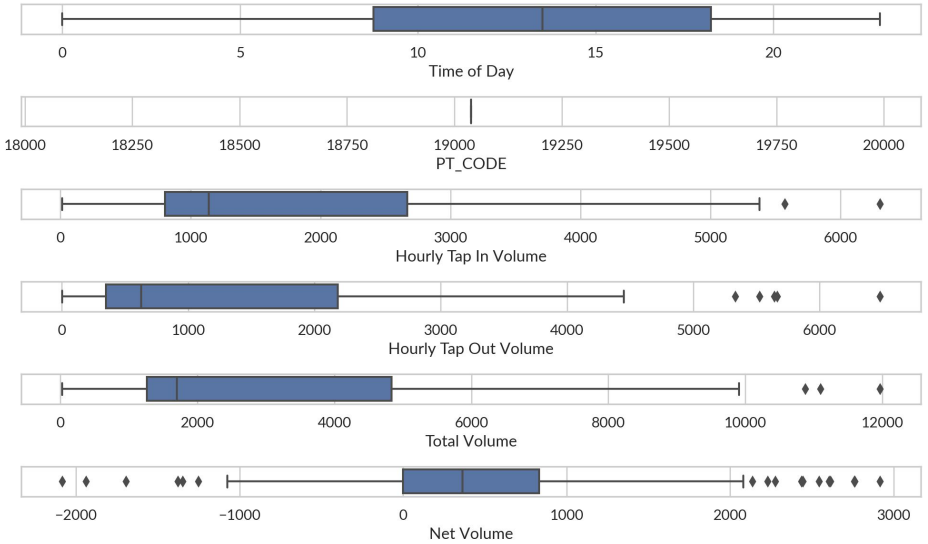
# Outliers

- Outliers are very similar to outliers in station volume dataset.
- It many of these outliers in the 100K and above range are bus interchanges.
- Other outliers appear to be facilities like Schools and Healthcare Organisations which would see more traffic than typical residential areas.



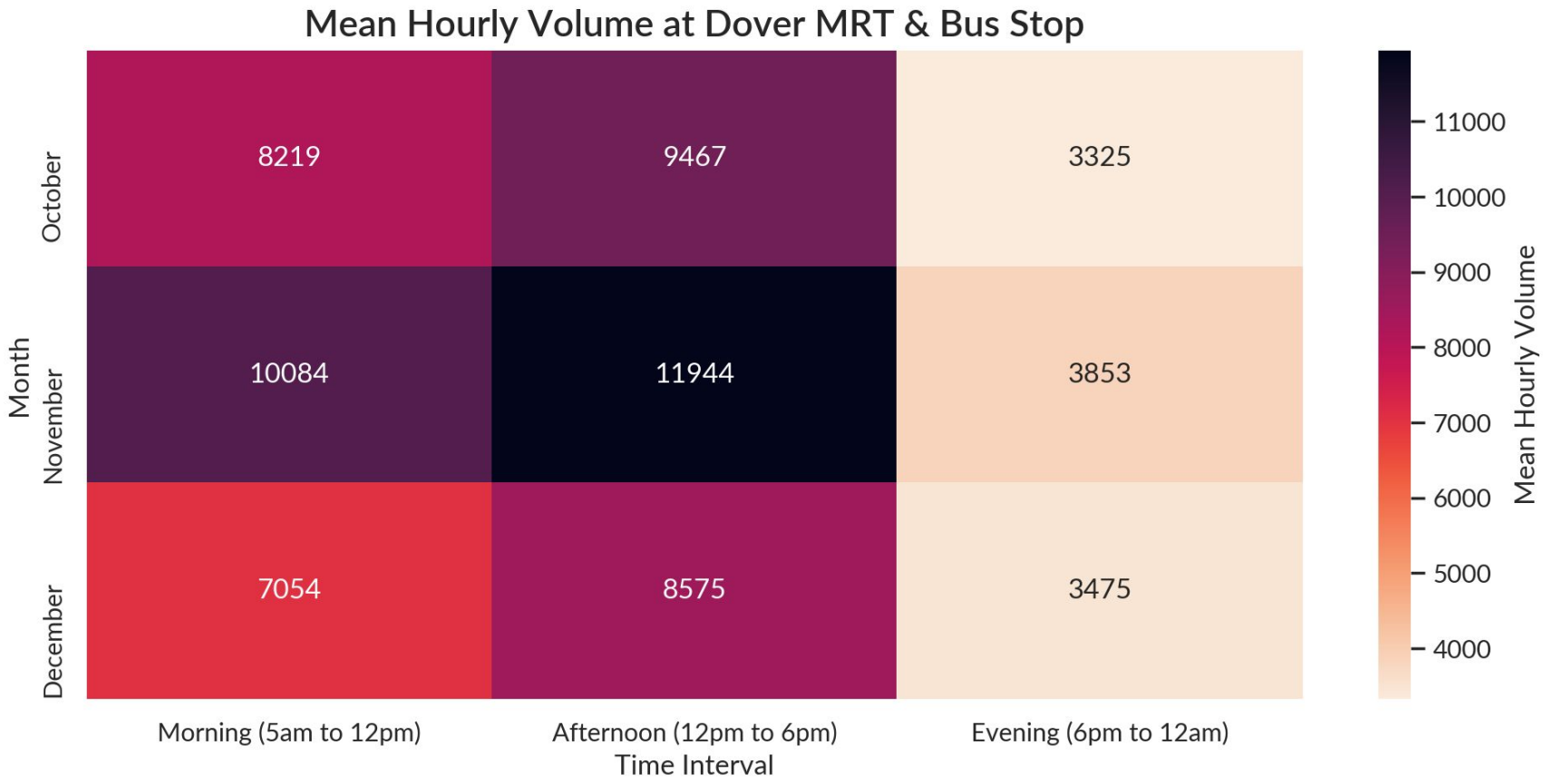
# Outliers

- When filtered down to Dover Bus Stop, many outliers are gone
- Remaining outliers are useful in telling us peak demand for Dover MRT Bus Stop, hence will not be removed.



```
sp_only = busmrt_volume[busmrt_volume["PT_CODE"].isin(["EW22", 19039])]
sp_only["Time Interval"] = pd.cut(sp_only["Time of Day"], [5,12,18,24], # Morn: 5 to 12, Afternoon: 12 to 6,
Evening: 6 to midnight
                          labels=["Morning", "Afternoon", "Evening"], include_lowest= True)
demand_df =pd.crosstab(index = sp_only["Year-Month"], columns=sp_only["Time Interval"], values = sp_only["Total
Volume"], aggfunc= np.mean)
fig, ax = plt.subplots(figsize=(10, 5), tight_layout = True)
sns.heatmap(demand_df, ax= ax, cmap="rocket_r", annot = True, fmt = ".0f", yticklabels = ["October" , "November",
"December"], cbar_kws = {
    "orientation": "vertical",
    "label" : "Mean Hourly Volume"
}, xticklabels = ["Morning (5am to 12pm)", "Afternoon (12pm to 6pm)", "Evening (6pm to 12am)"])
ax.set_title("Mean Hourly Volume at Dover MRT & Bus Stop", weight = "semibold", fontsize = 16)
ax.set_ylabel("Month")
fig.show()
fig.savefig("./plots/heatmap_volume.png", dpi = 200)
```

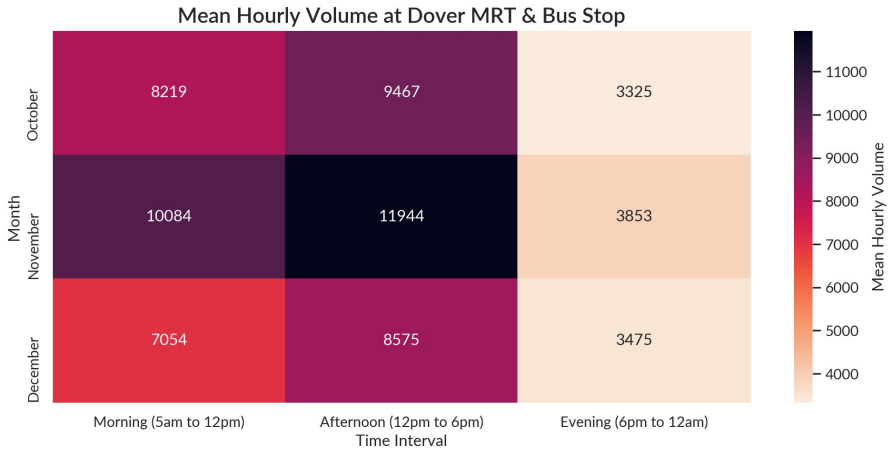
Graph 4 Code




How does the commuter demand differ by Month?

# Analysis

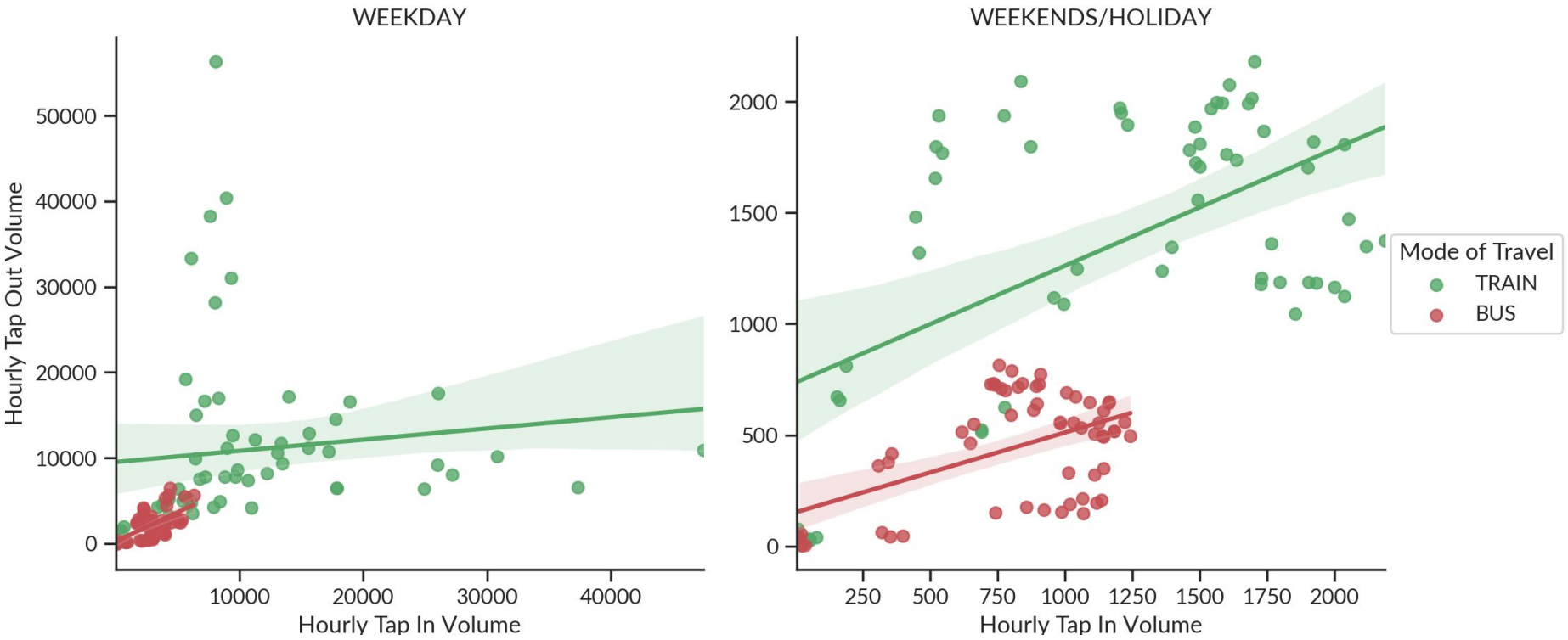
- Greatest demand in November across all time intervals (mean hourly volume of 10084 in morning is 1865 more than in October). Based on the SP academic calendar, this is during Term 3 (start of Sem 2).
- Lowest demand during December is likely due to there being an ongoing term vacation during that time which lasts until January. However, even with a vacation the mean hourly volume in the morning and afternoon is still relatively high at 7054 and 8575.
- Generally, the highest mean hourly volume occurs during the Afternoon Period, where the mean hourly volume increases from morning period  by about a 1543 people. (12pm to 6pm).



```
with sns.axes_style("ticks", { "font.family" : "Lato"}):
    lm_grid = sns.lmplot(x = "Hourly Tap In Volume", y = "Hourly Tap Out Volume", data = sp_only, hue="Mode of
Travel", col="Day Type", sharex = False, sharey= False, palette=["g", "r"], col_order = ["WEEKDAY",
"WEEKENDS/HOLIDAY"]) # plot multiple regression lines,
    lm_grid.fig.suptitle("Do Hourly Tap Ins Correlate With Tap Outs at Dover MRT & Bus Stop?", weight = "semibold")
    lm_grid.fig.subplots_adjust(top = 0.85) # adjust position of title to not overlap with graph
    lm_grid.set_titles(col_template ="{col_name}")
    lm_grid.legend.set_frame_on(True)
    lm_grid.fig.show()
lm_grid.fig.savefig("./plots/scatter_mrtbus.png", dpi = 200)
```

Graph 5 Code

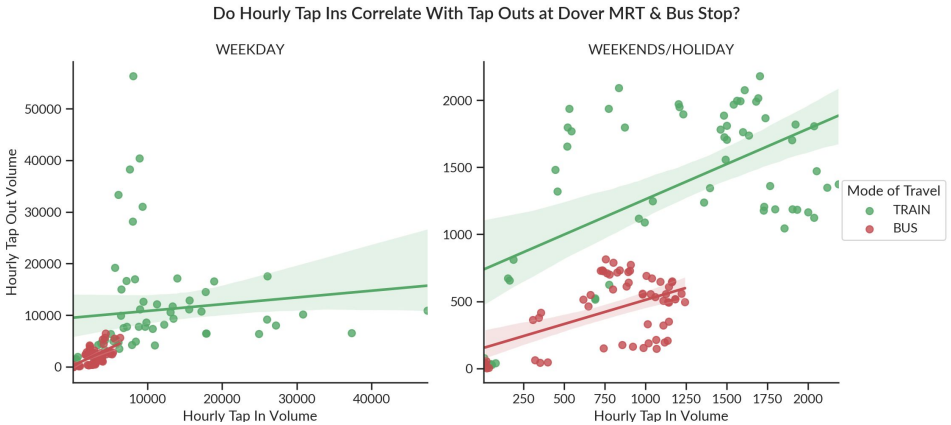
## Do Hourly Tap Ins Correlate With Tap Outs at Dover MRT & Bus Stop?




On Weekdays, relationship is weak, but stronger on Weekends.

# Insights

- On Weekdays, there is a moderate positive linear relationship between Tap in and Tap out for the Dover Bus Stop (R = 0.62), while the relationship between tap in and tap outs for Dover MRT is weak (R = 0.11).
- On Weekends/Holidays, this changes as for both MRT and Bus there is a moderate linear positive relationship between hourly tap ins and tap out volume (R = 0.57 (MRT) and R = 0.51 (Bus)) respectively.
- This likely reflects a change in the main user of the MRT station from students to local residents which now make up most passengers in the weekends.
- On a weekday, tap in and tap out volumes for Dover Bus Stop are concentrated in the range 60 to 7000. Tap in and tap out volumes for Dover MRT are more spread out with a greater range of 10 to 60000.
- On Weekends, the range of total volume for both MRT and Buses greatly decreases as the maximum tap in and out for train is now (2186, 2180) and (1243, 815) for bus. This suggests most of the volume in the station and bus stop is generated by SP students on weekdays.



**Should** the frequency of buses to SP be **increased**?

# Dataset 4: Bus Arrival

| # | Column | Non-Null Count | Dtype |
|---|---|---|---|
| 0 | ServiceNo | 1404 non-null | object |
| 1 | Operator | 1404 non-null | object |
| 2 | NextBus.OriginCode | 1404 non-null | float64 |
| 3 | NextBus.DestinationCode | 1404 non-null | float64 |
| 4 | NextBus.EstimatedArrival | 1404 non-null | object |
| 5 | NextBus.Latitude | 1404 non-null | float64 |
| 6 | NextBus.Longitude | 1404 non-null | float64 |
| 7 | NextBus.VisitNumber | 1404 non-null | float64 |
| 8 | NextBus.Load | 1404 non-null | object |
| 9 | NextBus.Feature | 1404 non-null | object |
| 10 | NextBus.Type | 1404 non-null | object |
| 11 | NextBus2.OriginCode | 1355 non-null | float64 |
| 12 | NextBus2.DestinationCode | 1355 non-null | float64 |
| 13 | NextBus2.EstimatedArrival | 1355 non-null | object |
| 14 | NextBus2.Latitude | 1355 non-null | float64 |
| 15 | NextBus2.Longitude | 1355 non-null | float64 |
| 16 | NextBus2.VisitNumber | 1355 non-null | float64 |
| 17 | NextBus2.Load | 1355 non-null | object |
| 18 | NextBus2.Feature | 1355 non-null | object |
| 19 | NextBus2.Type | 1355 non-null | object |
| 20 | NextBus3.OriginCode | 1292 non-null | float64 |
| 21 | NextBus3.DestinationCode | 1292 non-null | float64 |
| 22 | NextBus3.EstimatedArrival | 1292 non-null | object |
| 23 | NextBus3.Latitude | 1292 non-null | float64 |
| 24 | NextBus3.Longitude | 1292 non-null | float64 |
| 25 | NextBus3.VisitNumber | 1292 non-null | float64 |
| 26 | NextBus3.Load | 1292 non-null | object |
| 27 | NextBus3.Feature | 1292 non-null | object |
| 28 | NextBus3.Type | 1292 non-null | object |
| 29 | Timestamp | 1404 non-null | object |
| 30 | Bus Stop Code | 1404 non-null | object |

- Dataset retrieved from LTA Datamall API
- Real time bus arrival information on a bus stop
- Data collected covers October 2020 to December 2020 (Jan 2021 data unavailable)
- 1404 rows and 30 columns
- 112 rows with null values
- Data was sampled from 5 Feb to 11 Feb, and thus may not be completely representative of bus arrival.

```python
# bus_arrival.py
# The purpose of this script is to obtain bus arrival data for bus stops near Singapore Polytechnic.
import pandas as pd
import requests
import schedule # task scheduler
import time
import pytz
LTA_API_KEY = "iIjmlFM/RRGXIjsjY5KTnA=="
DataFrame = pd.core.frame.DataFrame
bus_stops = {
    "Dover MRT" : 19039,
    "Alongside Commonwealth Ave" : 19041
}
tz = pytz.timezone("Asia/Singapore")
def getArrivalInformation(stop_code: int, token : str) -> DataFrame:
    try:
        req = requests.get("http://datamall2.mytransport.sg/ltaodataservice/BusArrivalv2", params = {
            "BusStopCode" : stop_code
        }, headers = { "AccountKey" : token })
        print(req)
        req.raise_for_status()
        data = req.json()
        df = pd.json_normalize(data, "Services") # flatten json to dataframe
        df["Timestamp"] = pd.Timestamp.now(tz).round("min")
        df["Bus Stop Code"] = stop_code
        return df
    except Exception as e:
        print(e)
        print(f"Sorry! There was some issue getting data at {pd.Timestamp.now(tz).round('30min')}")
        return None
def job():
    print("job running")
    output_df = None # declare final output
    for code in bus_stops.values(): # for each bus stop code
        data = getArrivalInformation(code, LTA_API_KEY) # get arrival information
        if data is None: # if info is unavailable (error)
              output_df = None # then final output is nothing
              pass
        if output_df is None:
            output_df = data # dataframe or none
        else:
            output_df = output_df.append(data, ignore_index = True) # append
    if output_df is not None:
        output_df.to_csv(f"./bus_arrival/bus_arrival_{pd.Timestamp.now(tz)}.csv".replace(":", "-"))
    else:
        print("Sorry! We could not create csv file")
schedule.every(30).minutes.do(job)
job()
while True:
    schedule.run_pending()
    time.sleep(1)
```

# Bus Arrival Collection Script

# Data Preprocessing

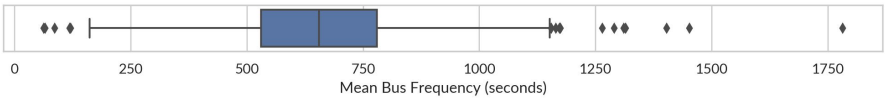
1. All csv files loaded using pd.read_csv, and concatenated together using pd.concat into a single DataFrame
2. Duplicate Service No. merged by converting Service No. col to type string (.astype(str))
3. Unnecessary columns dropped using .drop()
4. All timestamp data converted from string into a local timestamp. (.astype("datetime64").dt.tz_localize("UTC").dt.tz_convert("Asia/Singapore"))
5. dt.hour and dt.day_name() used to create time of day and day columns
6. .diff(periods = 1, axis = 1).apply(lambda x : x.dt.seconds).mean(axis = 1) used to obtain estimated bus frequency

$$\overline{BusFreq} = \frac{1}{N-1}\sum_{i=2}^{N}(Estimated\ Arrival\ Time\ for\ Bus\ i - Estimated\ Arrival\ Time\ for\ Bus\ (i-1))\ where\ N = Number\ of\ Buses\ Ahead$$

7. Missing values checked and counted using .isna().sum()

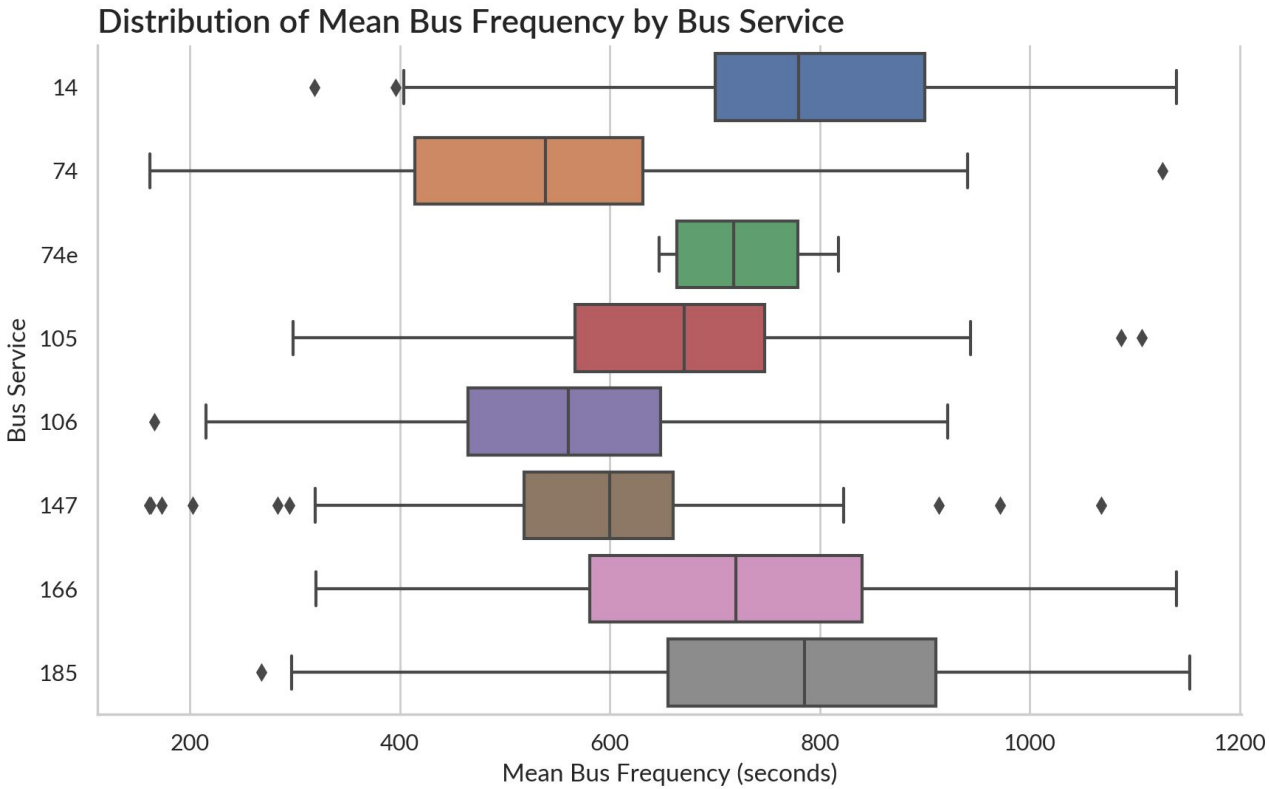
# Outliers

- Outliers in this dataset were removed.
- They generally represented a trend of bus bunching, where due to delays somewhere along the route, two buses of the same service arrive almost at the same time.
- The presence of it tells us that some buses are not able to keep to schedule.
- However, to have an accurate estimate of actual mean bus frequency, it is necessary to remove these outliers.
- removeOutliers function was used to remove the outliers



```
bus_arrival_df_no_outliers = bus_arrival_df.copy()
bus_arrival_df_no_outliers["Mean Bus Frequency (seconds)"] = removeOutliers(bus_arrival_df, ["Mean Bus Frequency
(seconds)"])
fig, ax = plt.subplots(figsize = (10, 6))
sns.boxplot(x = "Mean Bus Frequency (seconds)",y="ServiceNo", data = bus_arrival_df_no_outliers, ax= ax, order =
['14', '74', '74e', '105', '106', '147', '166', '185'])
ax.set_title("Distribution of Mean Bus Frequency by Bus Service", loc="left", weight="semibold", fontsize = 16)
ax.set_ylabel("Bus Service")
sns.despine()
fig.show()
fig.savefig("./plots/bus_freq_box.png", dpi = 200)
```

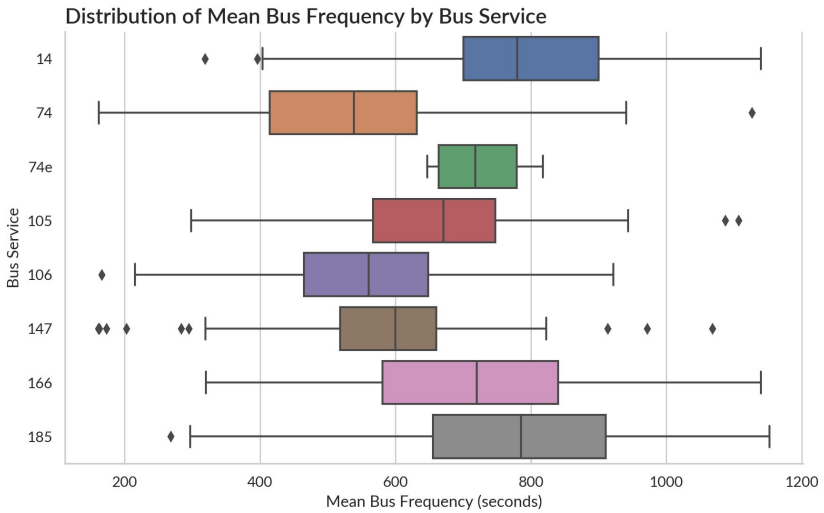
Chart 6 Code




What is the typical distribution of bus frequencies by Service No?

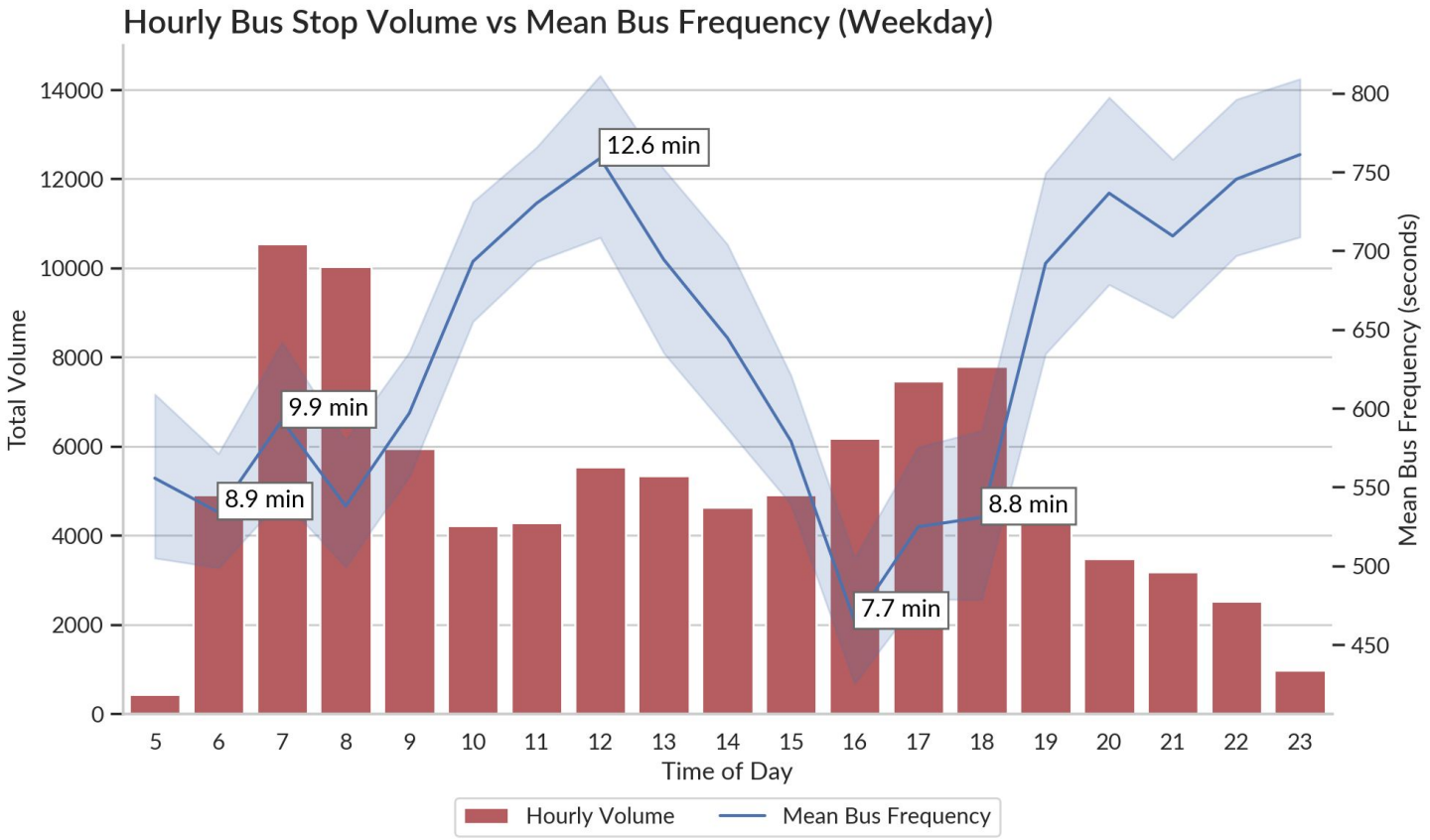
# Analysis

- Distribution of mean bus frequency tend to be centered around 550 to 800 seconds. Means mean bus frequency to SP is about 9 to 13 minutes.
- Based on LTA's promise that all buses should have a frequency of 15 minutes (900 seconds) or less during peak periods, we observe that services 185, 166 and 14 which have an upper quartile close to 900, may need to improve their bus frequency to reduce chance of overcrowding.
- Service 74e has the lowest IQR (around 1 to 2 minute). 74e is an express service, meaning it travels faster by having less bus stops & travelling on expressways. This allows for the greater consistency in bus frequency shown here.
- Evidence of bus bunching as shown by outliers; Means that some buses are arriving right after another, causing the wait for the next bus to be extremely long.



```
# Subset to 5am and after
bus_arrival_filtered = bus_arrival_df_no_outliers[(bus_arrival_df_no_outliers["Time of Day"] >= 5) &
(~(bus_arrival_df_no_outliers["Day of Week"].isin(["Saturday", "Sunday"])))]
bus_volume_sp_filtered = bus_only[(bus_only["Time of Day"] >= 5) & (bus_only["Day Type"] == "WEEKDAY")]

# Plotting
fig, ax = plt.subplots(figsize = (10, 6), tight_layout = True)
ax2 = ax.twinx()
# ax.set_xticks(bus_arrival_filtered["Time of Day"].unique())
line_plot = sns.lineplot(x= bus_arrival_filtered["Time of Day"] - 5,y= "Mean Bus Frequency (seconds)",data =
bus_arrival_filtered, ax = ax2, ci = 95, label="Mean Bus Frequency", legend = False) # -5 as bar plot in seaborn
always starts axis from 0, meaning first value (even if it is a number), will always be 0. Thus to align the
lineplot, I need to -5 from all values, so the smallest value (5) is read as a 0.
sns.barplot(x = "Time of Day", y = "Total Volume", data = bus_volume_sp_filtered, palette=["r"], ax = ax, ci =
False, label="Hourly Volume")

# Annotation and Visuals
ax.set_title("Hourly Bus Stop Volume vs Mean Bus Frequency (Weekday)", loc = "left",weight = "semibold", fontsize =
16)
ax2.grid(None)
ax.set_ylim(0, 15000) # make lineplot higher up so they don't intersect
sns.despine()

linesAx, labelsAx = ax.get_legend_handles_labels()
linesAx2, labelsAx2 = ax2.get_legend_handles_labels()
ax.legend(lines + lines2, labels + labels2, bbox_to_anchor=(0.5, -0.2), loc="lower center", ncol = 2)
line_annot = [ax2.text(s = f"{y / 60:.1f} min", x= x - 4.9, y = y + 3, color = "black", bbox = bbox) for x, y in
bus_arrival_filtered.groupby("Time of Day").mean()["Mean Bus Frequency (seconds)"].iloc[[1, 2, 7, 11,
13]].iteritems()] # Annotate bus frequency
fig.show()
fig.savefig("./plots/volume_vs_freq.png", dpi = 200)
```

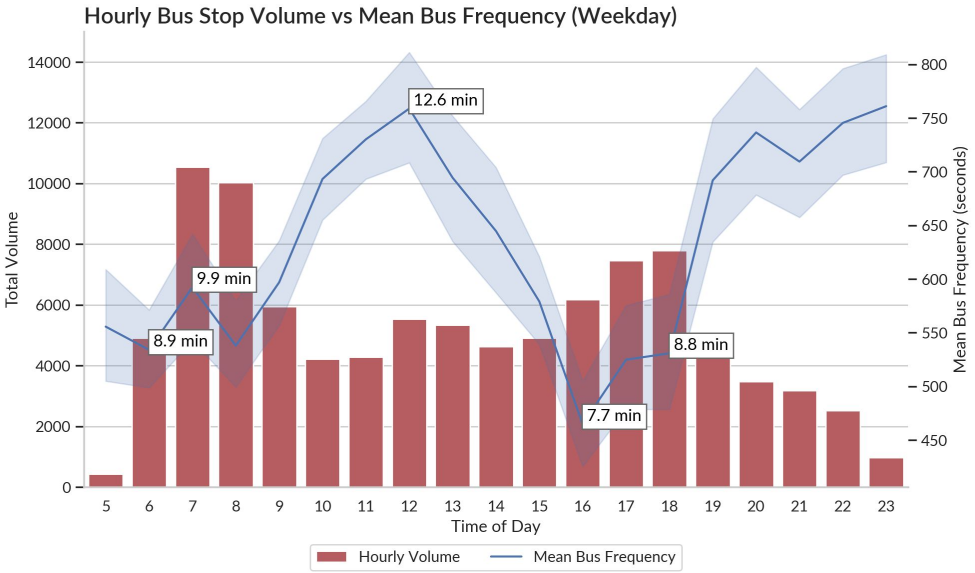
## Chart 7 Code




Are bus frequencies keeping up with the demand for buses?

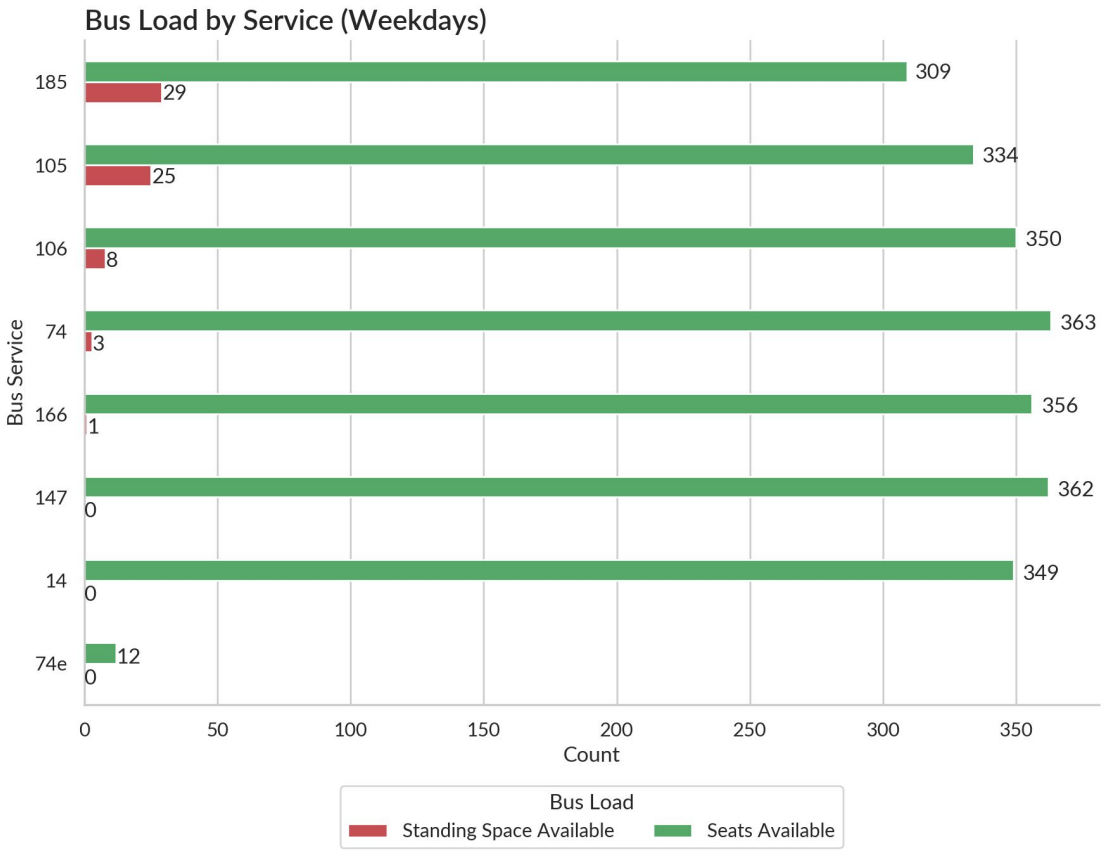
# Analysis

- Two peak periods where bus volume is highest, 6am to 9pm, 4pm to 7pm. Highest bus stop volume at 7am period (~10549 people). In these periods, mean frequency ranges from around 8 to 10 minutes. Based on LTA, this frequency is generally good for peak hours as benchmark is 15 minutes and less.
- Those taking buses have slightly different commuter patterns as morning peak hour (7am) is one hour earlier than the MRT peak hour (8am), while evening peak hour is an hour later than the MRT.
- Bus Services may have difficulties handling the 7am period crowd, as mean bus frequency increased by 1 minute during this time.



```
bus_arrival_filtered_load = bus_arrival_with_na[~(bus_arrival_with_na["Day of Week"].isin(["Saturday", "Sunday"]))]
# count those where there are na values
load_df = bus_arrival_filtered_load.melt(id_vars ="ServiceNo",value_vars=["NextBus.Load", "NextBus2.Load",
"NextBus3.Load"])
load_df = pd.crosstab(index = load_df["ServiceNo"], columns = load_df["value"]).sort_values(["SDA", "SEA"],
ascending = True)
load_df.rename(columns = {
    "SDA" : "Standing Space Available",
    "SEA" : "Seats Available"
}, inplace = True)
fig, ax = plt.subplots(figsize = (9, 7), tight_layout = True)
load_df.plot(kind = "barh", ax = ax, stacked = False, color = ["r", "g"])
ax.set_title("Bus Load by Service", loc = "left", weight = "semibold", fontsize = 16)
ax.set_ylabel("Bus Service")
ax.set_xlabel("Count")
ax.grid(False, axis = "y")
ax.legend(title="Bus Load", bbox_to_anchor=(0.5, -0.23), loc="lower center", ncol = 2)
sns.despine()
annot = [ax.annotate(f"{p.get_width()}", (p.get_width() * 1.01, p.get_y() * 1.005)) for p in ax.patches]
fig.show()
fig.savefig("./plots/load_count.png", dpi = 200)
```

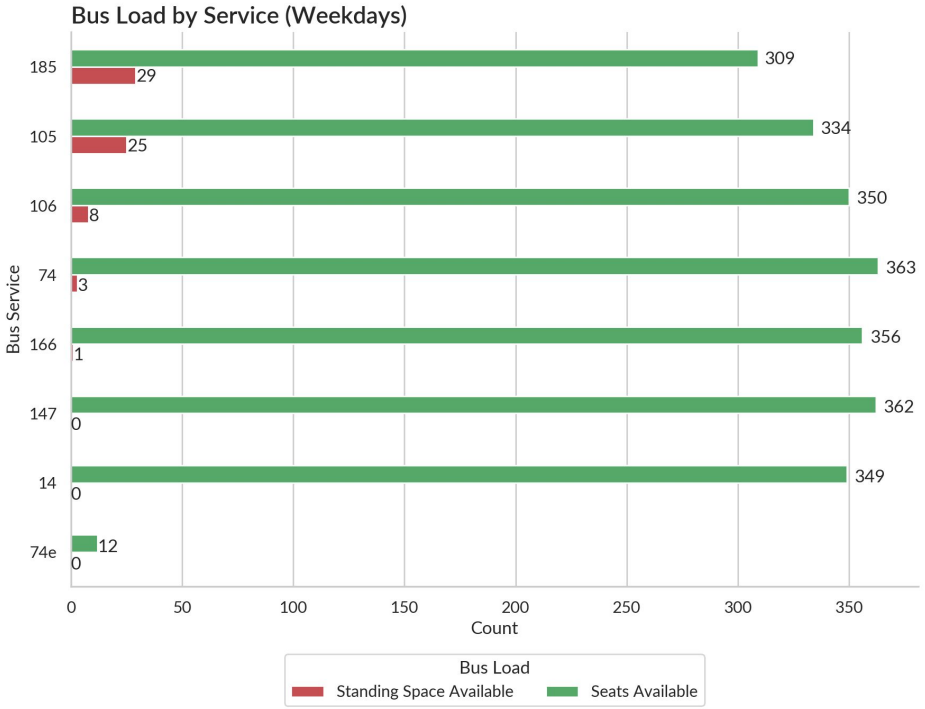
Graph 8 Code




## Which Bus Services Face the Heaviest Loads?

# Analysis

- Overall, some bus services like 147, and 14, as based on the sample data collected, the load is always low. (0 occurrences of no seats available). These three routes go from East Singapore to West Singapore. Eg. 147 goes from Hougang to Clementi, 14 from Bedok to Clementi.
- The Bus Services with the most occurrences of high load (standing space available only) during weekdays are Service No. 185 (9.39%), 105 (7.49%) and 106 (2.29%). Based on Moovit route info, These three routes cover Western to Central Singapore. For instance 185 covers Pioneer to Buona Vista.
- This suggests the demographic of people taking a bus to SP are mostly made up of people staying nearby (Western/Central area). This would make sense as those living further away would reach SP faster via the MRT.



# Conclusion

# Conclusion

- Peak demand for travel to and from SP occurs in the morning at 6am to 10am and in the evening from 4pm to 7pm. Morning peak hour sits between peak hour for Jurong East and Raffles Place.
- On Weekdays, net volume of Dover MRT is usually negative in the morning, and positive in the evening.
- Most travel to SP by MRT instead of bus; Those that travel by bus seem to be those living in Central/West side of Singapore.
- Net demand differs by month; For months in more demanding academic terms, the increase in mean hourly total volume can be >1.5k
- Current bus frequencies are above average, but can be more consistent.

# Recommendations

## For the MRT:

- SP can look into class timetables, to try and schedule classes such as to **reduce number of 9am classes** (causing peak 8am demand). This will **spread out the demand** reducing overcrowding.
- SP can also consider doing more home based learning in months closer to end of semester.

## For Bus Services:

- SP can look into **operating shuttle buses** which can run at higher frequency than typical buses serving SP (8 minutes per bus at peak hours) in the morning from 7am to 9am and in the afternoon from 5pm to 6pm.
- SP can recommend that LTA look into how bus frequency could be made more consistent during 7am time period. A possible avenue would be to try and reduce occurance of bus bunching which was observed.

# Data Sources & References

# Dataset Sources


- data.gov.sg. 2021. Public Transport Utilisation - Average Daily Public Transport Ridership-Data.gov.sg. [online] Available at: <https://data.gov.sg/dataset/public-transport-utilisation-average-public-transport-ridership> [Accessed 11 February 2021].
- LTA Datamall APIs: https://datamall.lta.gov.sg/content/datamall/en/dynamic-data.html
    - LTA Datamall. n.d. API: Passenger Volume by Train Stations. [online] Available at: <http://datamall2.mytransport.sg/ltaodataservice/PV/Train> [Accessed 11 February 2021].
    - LTA Datamall. n.d. API: Passenger Volume by Bus Stops. [online] Available at: <http://datamall2.mytransport.sg/ltaodataservice/PV/Bus> [Accessed 11 February 2021].
    - LTA Datamall. n.d. API: Bus Arrival. [online] Available at: <http://datamall2.mytransport.sg/ltaodataservice/BusArrivalv2> [Accessed 11 February 2021].

# References


- https://www.sp.edu.sg/sp/about-sp/visit-us/directions-to-sp
- https://landtransportguru.net/bus-bunching/
- https://www.singstat.gov.sg/find-data/search-by-theme/population/population-and-population-structure/latest-data
- https://sg.news.yahoo.com/lta-to-improve-17mrt-stations-in-2016-084430408.html
- https://www.smrt.com.sg/Portals/0/PDFs/Annual%20Reports/2004_AR.pdf
- https://moovitapp.com/index/en-gb/public_transportation-line-185-Singapore_%E6%96%B0%E5%8A%A0%E5%9D%A1-1678-775180-589151-0
- https://moovitapp.com/index/en-gb/public_transportation-line-105-Singapore_%E6%96%B0%E5%8A%A0%E5%9D%A1-1678-775180-589115-0
- https://moovitapp.com/index/en-gb/public_transportation-line-106-Singapore_%E6%96%B0%E5%8A%A0%E5%9D%A1-1678-873544-589258-0
- https://www.straitstimes.com/singapore/transport/bus-frequency-improved-buses-also-less-congested-lta
- https://moovitapp.com/index/en-gb/public_transportation-line-147-Singapore_%E6%96%B0%E5%8A%A0%E5%9D%A1-1678-775180-589132-0
- https://www.straitstimes.com/singapore/transport/reduced-frequency-of-trains-leads-to-crowding-on-some
- https://www.sp.edu.sg/sp/student-services/academic-calendar
- https://www.justlanded.com/english/Singapore/Singapore-Guide/Jobs/Working-Conditions