MTA Daily

July 23, 2024

[270]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from encodings.aliases import aliases

%matplotlib inline
#the matplotlib plots will appear directly below the cell
```

[271]:
```
#to find encodings that work

#Below line creates a set of all available encodings
alias_values=set(aliases.values())

for encoding in set(aliases.values()):
    try:
        df=pd.read_csv("mta.csv",nrows=10, encoding=encoding) #read in 10 lines␣
 ↪for faster read
        print('successful', encoding)
    except:
        pass
```

```
successful cp950
successful euc_kr
successful cp863
successful iso8859_13
successful johab
successful euc_jis_2004
successful ptcp154
successful utf_7
successful cp1140
successful cp1250
successful utf_16_be
successful shift_jis
successful cp949
successful gbk
successful shift_jisx0213
successful cp932
```

```
successful iso8859_15
successful cp850
successful euc_jisx0213
successful cp775
successful cp1125
successful iso8859_10
successful kz1048
successful cp500
successful cp861
successful cp869
successful shift_jis_2004
successful cp1252
successful cp862
successful iso2022_jp_2
successful cp1254
successful koi8_r
successful iso2022_jp_2004
successful iso2022_kr
successful hz
successful cp1257
successful big5
successful utf_16_le
successful cp1251
successful cp855
successful iso8859_7
successful hp_roman8
successful iso8859_9
successful ascii
successful cp273
successful cp852
successful iso8859_14
successful cp860
successful gb2312
successful utf_8
successful iso2022_jp_3
successful iso8859_2
successful mac_turkish
successful cp1258
successful mac_greek
successful cp866
successful latin_1
successful iso8859_6
successful cp1256
successful mac_iceland
successful big5hkscs
successful cp1026
successful cp037
successful cp857
```

```
successful gb18030
successful cp858
successful cp864
successful iso2022_jp
successful mac_latin2
successful mac_roman
successful iso8859_8
successful cp1253
successful mac_cyrillic
successful euc_jp
successful tis_620
successful cp1255
successful iso8859_16
successful iso8859_4
successful iso8859_11
successful cp437
successful iso2022_jp_1
successful iso8859_5
successful iso2022_jp_ext
successful iso8859_3
successful cp865
```

[272]:
```
#Read in the crime.csv file and use the timestamp as a datetime index
mta = pd.read_csv("mta.csv", encoding="iso8859_13")
```

[273]:
```
mta.head()
```

[273]:

	Date	Subways: Total Estimated Ridership	Subways: % of Comparable Pre-Pandemic Day	Buses: Total Estimated Ridership	Buses: % of Comparable Pre-Pandemic Day
0	07/18/2024	NaN	NaN	NaN	NaN
1	07/17/2024	3445598.0	0.65	1087975.0	0.53
2	07/16/2024	3685030.0	0.70	1198969.0	0.58
3	07/15/2024	3664346.0	0.69	1227459.0	0.59
4	07/14/2024	1773318.0	0.76	610618.0	0.56

	LIRR: Total Estimated Ridership	LIRR: % of Comparable Pre-Pandemic Day	Metro-North: Total Estimated Ridership	Metro-North: % of Comparable Pre-Pandemic Day	Access-A-Ride: Total Scheduled Trips	Access-A-Ride: % of Comparable Pre-Pandemic Day	Bridges and Tunnels: Total Traffic	Bridges and Tunnels: % of Comparable Pre-Pandemic Day
0	241785.0	0.76	211370	0.75	36119	1.27	NaN	NaN
1	248424.0	0.78	214137	0.76	36547	1.29	956017.0	0.99
2	249559.0	0.79	220621	0.78	35468	1.25	943841.0	0.98
3	233209.0	0.74	202008	0.71	32763	1.16	950682.0	
4	119283.0	1.14	97038	0.91	22682	1.37	947148.0	

```
3                                                 0.99
4                                                 1.07

   Staten Island Railway: Total Estimated Ridership  \
0                                                 NaN
1                                              6842.0
2                                              6968.0
3                                              6803.0
4                                              2117.0

   Staten Island Railway: % of Comparable Pre-Pandemic Day
0                                                  NaN
1                                                 0.50
2                                                 0.51
3                                                 0.49
4                                                 0.59
```

[274]:
```
mta.shape #checking the shape of the data as it has 319,073 rows and 17 columns
```

[274]:
```
(1601, 15)
```

[275]:
```
mta.duplicated().sum() #number of duplicated rows
```

[275]:
```
0
```

[276]:
```
mta.info() #summary info about the dataframe
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1601 entries, 0 to 1600
Data columns (total 15 columns):
 #   Column                                            Non-Null Count
Dtype
---  ------                                            --------------
-----
 0   Date                                              1601 non-null
object
 1   Subways: Total Estimated Ridership                1600 non-null
float64
 2   Subways: % of Comparable Pre-Pandemic Day         1600 non-null
float64
 3   Buses: Total Estimated Ridership                  1600 non-null
float64
 4   Buses: % of Comparable Pre-Pandemic Day           1600 non-null
float64
 5   LIRR: Total Estimated Ridership                   1600 non-null
float64
 6   LIRR: % of Comparable Pre-Pandemic Day            1600 non-null
```

```
float64
 7   Metro-North: Total Estimated Ridership                    1601 non-null
int64
 8   Metro-North: % of Comparable Pre-Pandemic Day             1601 non-null
float64
 9   Access-A-Ride: Total Scheduled Trips                      1601 non-null
int64
 10  Access-A-Ride: % of Comparable Pre-Pandemic Day           1601 non-null
float64
 11  Bridges and Tunnels: Total Traffic                        1600 non-null
float64
 12  Bridges and Tunnels: % of Comparable Pre-Pandemic Day     1600 non-null
float64
 13  Staten Island Railway: Total Estimated Ridership          1568 non-null
float64
 14  Staten Island Railway: % of Comparable Pre-Pandemic Day   1568 non-null
float64
dtypes: float64(12), int64(2), object(1)
memory usage: 187.7+ KB
```

[ ]:
```
# Renaming the columns
```

[ ]:
```
mta.rename(columns={'LIRR: Total Estimated Ridership': 'Lirr'}, inplace=True)
```

[321]:
```
mta.head()
```

[321]:

	Date	Subway	Lirr	Month	Year
0	2024-07-18	NaN	241785.0	July	2024
1	2024-07-17	3445598.0	248424.0	July	2024
2	2024-07-16	3685030.0	249559.0	July	2024
3	2024-07-15	3664346.0	233209.0	July	2024
4	2024-07-14	1773318.0	119283.0	July	2024

[322]:
```
#Isolate LIRR
# List of columns to keep
columns_to_keep = ['Date', 'Lirr' ]

# Keep only the selected columns
mta = mta[columns_to_keep]

print(mta)
```

```
          Date      Lirr
0   2024-07-18  241785.0
1   2024-07-17  248424.0
2   2024-07-16  249559.0
3   2024-07-15  233209.0
4   2024-07-14  119283.0
```

```
...          ...       ...
1596 2020-03-04  311662.0
1597 2020-03-03  319727.0
1598 2020-03-02  321569.0
1599 2020-03-01       NaN
1600 2023-06-19  196645.0

[1601 rows x 2 columns]
```

[323]:
```
#checking for columns with missing values
mta.columns[np.sum(mta.isnull()) !=0]
```

```
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-
packages/numpy/core/fromnumeric.py:84: FutureWarning: The behavior of
DataFrame.sum with axis=None is deprecated, in a future version this will reduce
over both axes and return a scalar. To retain the old behavior, pass axis=0 (or
do not pass axis)
  return reduction(axis=axis, out=out, **passkwargs)
```

[323]:
```
Index(['Lirr'], dtype='object')
```

[281]:
```
# Convert the Date column to datetime
mta['Date'] = pd.to_datetime(mta['Date'])

print(mta.dtypes)
```

```
Date      datetime64[ns]
Subway           float64
Lirr             float64
dtype: object
```

[355]:
```
### HAS AVERAGE MONTHLY RIDERSHIP GONE UP OR DOWN SINCE 2020?
```

[356]:
```
# Aggregate the ridership by date
daily_ridership = mta.groupby('Date')['Lirr'].sum()
```

[357]:
```
mta.head()
```

[357]:

	Date	Lirr	Month	Year
0	2024-07-18	241785.0	July	2024
1	2024-07-17	248424.0	July	2024
2	2024-07-16	249559.0	July	2024
3	2024-07-15	233209.0	July	2024
4	2024-07-14	119283.0	July	2024

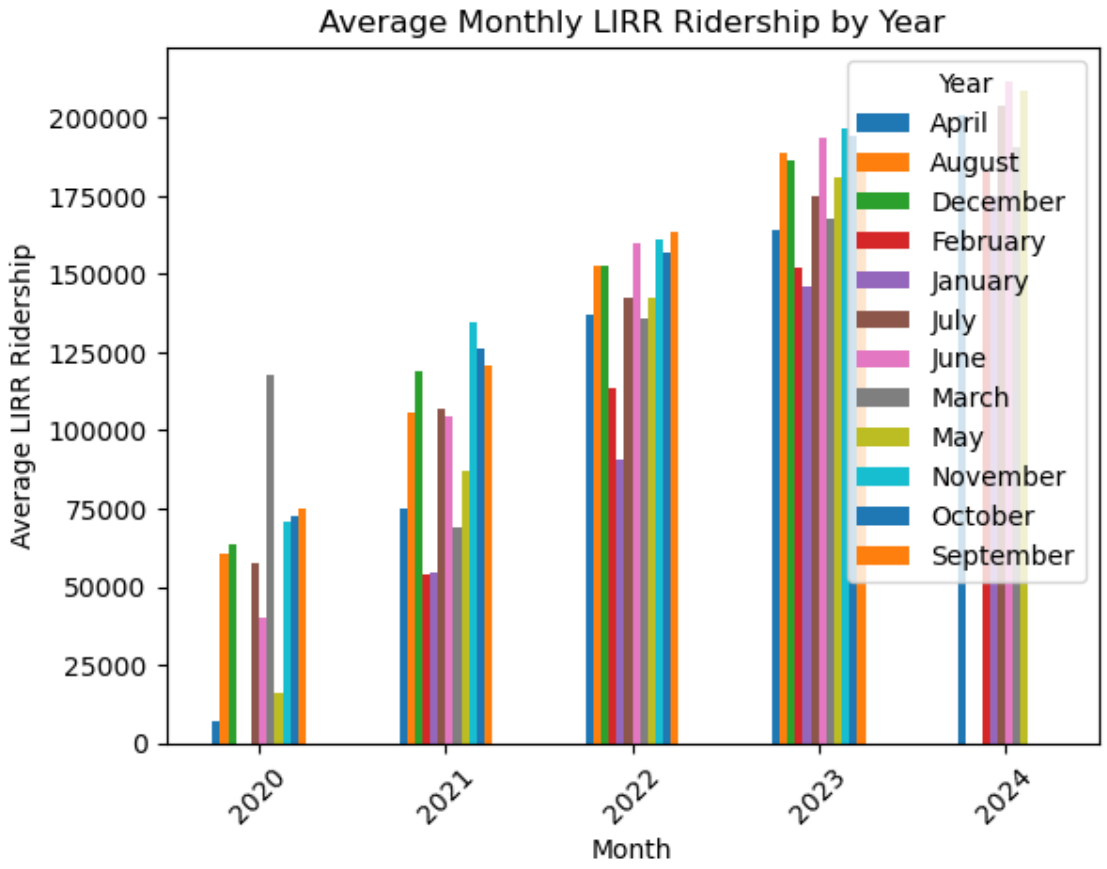
[358]:
```
# Extract month and year from the 'Date' column
mta['Month'] = mta['Date'].dt.month_name()
mta['Year'] = mta['Date'].dt.year
```

```
/tmp/ipykernel_226/257613710.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  mta['Month'] = mta['Date'].dt.month_name()
/tmp/ipykernel_226/257613710.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  mta['Year'] = mta['Date'].dt.year
```

[359]:
```
# Group by month and year, then calculate the average LIRR ridership
monthly_ridership = mta.groupby(['Year', 'Month'])['Lirr'].mean().unstack()
```

[360]:
```
# Plot the average monthly ridership with different colors for each year
plt.figure(figsize=(12, 8))
monthly_ridership.plot(kind='bar', color=plt.cm.tab10.colors)  # Use a colormap␣
 ↪to differentiate years
plt.xlabel('Month')
plt.ylabel('Average LIRR Ridership')
plt.title('Average Monthly LIRR Ridership by Year')
plt.xticks(rotation=45)
plt.legend(title='Year')
plt.show()
```

```
<Figure size 1200x800 with 0 Axes>
```



[361]:
```
# Sort the months in the correct order
monthly_ridership = monthly_ridership.reindex(columns=[
    'January', 'February', 'March', 'April', 'May', 'June',
    'July', 'August', 'September', 'October', 'November', 'December'
])
```

[362]:
```
mta.head()
```

[362]:

	Date	Lirr	Month	Year
0	2024-07-18	241785.0	July	2024
1	2024-07-17	248424.0	July	2024
2	2024-07-16	249559.0	July	2024
3	2024-07-15	233209.0	July	2024
4	2024-07-14	119283.0	July	2024

[363]:
```
#Average ridership has been increasing steadily since 2020 to 2024
```

[364]:
```
### WHAT WAS BUSIEST MONTH IN 2023?
```

[365]:
```
# Filter for the year 2023
mta_2023 = mta[mta['Date'].dt.year == 2023]
```

[366]:
```
# Extract month names
mta_2023['Month'] = mta_2023['Date'].dt.month_name()
```

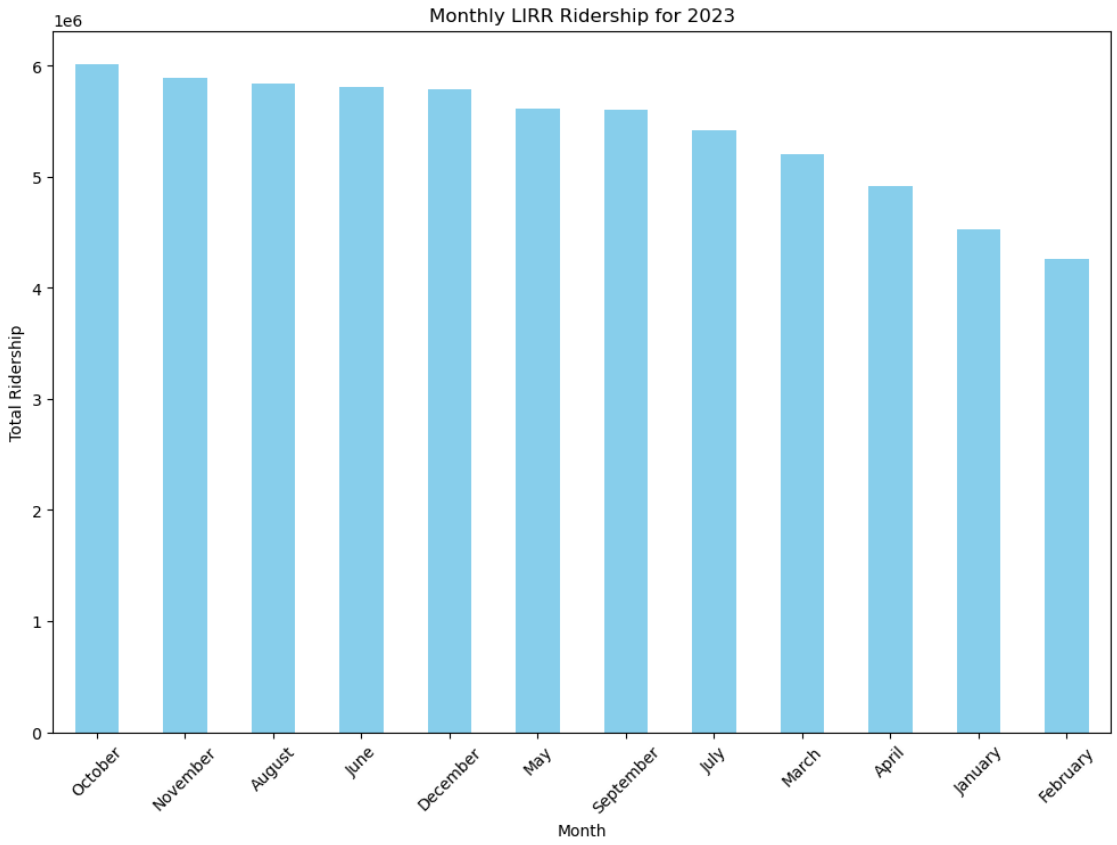
```
/tmp/ipykernel_226/3551488563.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  mta_2023['Month'] = mta_2023['Date'].dt.month_name()
```

[367]:
```
# Aggregate ridership by month
monthly_ridership_2023 = mta_2023.groupby('Month')['Lirr'].sum()
```

[368]:
```
# Ensure it's a Series (1D array)
if isinstance(monthly_ridership_2023, pd.DataFrame):
    monthly_ridership_2023 = monthly_ridership_2023.squeeze()

# Sort by ridership from highest to lowest
monthly_ridership_2023 = monthly_ridership_2023.sort_values(ascending=False)
```

[369]:
```
# Check if the Series has any data
if monthly_ridership_2023.empty:
    raise ValueError("monthly_ridership_2023 is empty or not a 1D Series")
```

[370]:
```
# Plot bar chart
plt.figure(figsize=(12, 8))
monthly_ridership_2023.plot(kind='bar', color='skyblue')
plt.xlabel('Month')
plt.ylabel('Total Ridership')
plt.title('Monthly LIRR Ridership for 2023')
plt.xticks(rotation=45)
plt.show()
```



[371]:
```
#CONCLUSION
# The above bar chart shows that October is the busiest month in 2023 with␣
 ↪February have the least traffic.
```

[372]:
```
#### ON AVERAGE, WHAT DAYS OF THE WEEK WERE THE BUSIEST IN 2023?
```

[373]:
```
# Filter for the year 2023
mta_2023 = mta[mta['Date'].dt.year == 2023]
```

[374]:
```
# Extract day of the week names
mta_2023['DayOfWeek'] = mta_2023['Date'].dt.day_name()
```

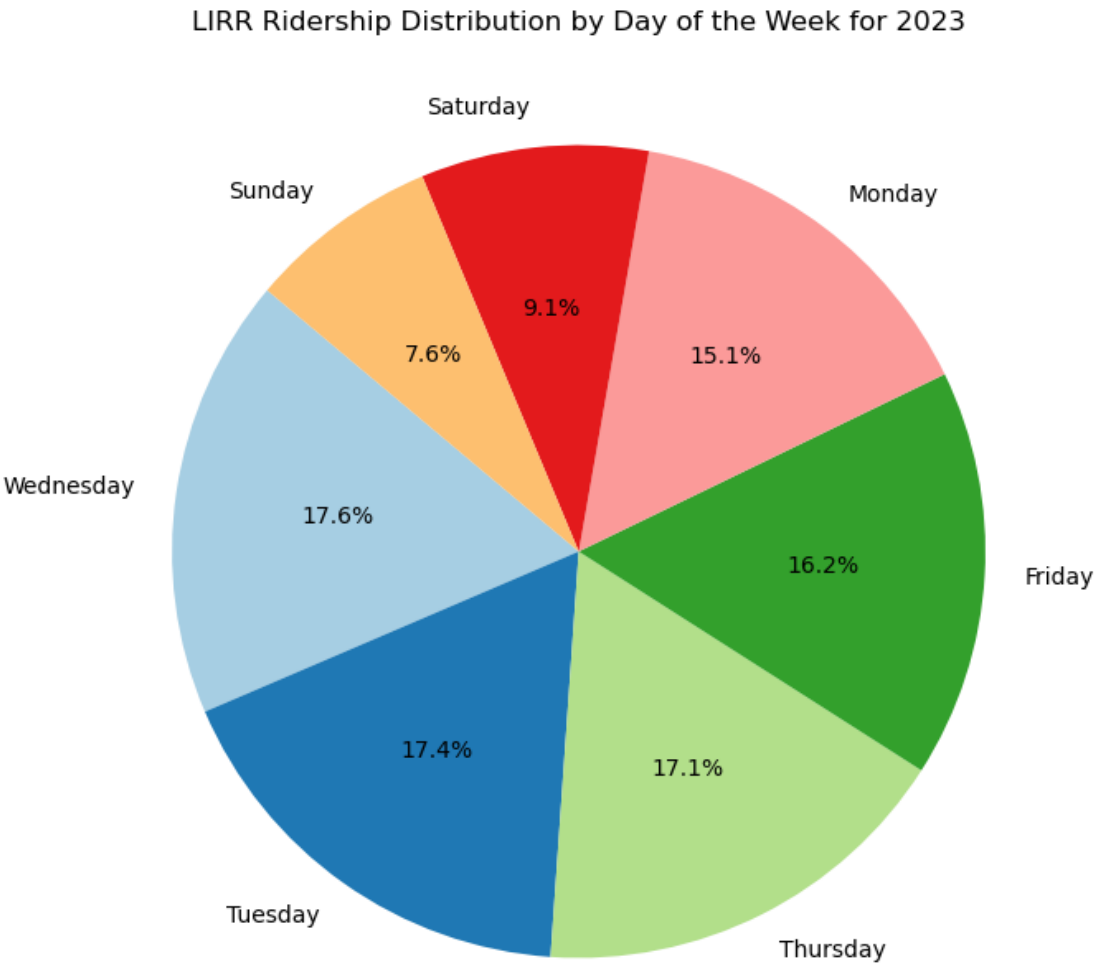
```
/tmp/ipykernel_226/318176216.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  mta_2023['DayOfWeek'] = mta_2023['Date'].dt.day_name()
```

[375]:
```
# Aggregate ridership by day of the week
daily_ridership_2023 = mta_2023.groupby('DayOfWeek')['Lirr'].sum()
```

[376]:
```
# Reorder days of the week to ensure correct sequence
daily_ridership_2023 = daily_ridership_2023.reindex([
    'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday'
])
```

[377]:
```
# Sort by ridership from highest to lowest
daily_ridership_2023 = daily_ridership_2023.sort_values(ascending=False)
```

[378]:
```
# Plot pie chart
plt.figure(figsize=(10, 8))
plt.pie(daily_ridership_2023, labels=daily_ridership_2023.index, autopct='%1.
↪1f%%', startangle=140, colors=plt.cm.Paired.colors)
plt.title('LIRR Ridership Distribution by Day of the Week for 2023')
plt.show()
```



[379]:
```
#CONCLUSION
#The above chart shows that Wednesdays are the busiest with Sunday having the␣
 ↪least volume in 2023.
```

[ ]:
```

```

[387]:
```
mta
```

[387]:

	Date	Lirr	Month	Year
0	2024-07-18	241785.0	July	2024
1	2024-07-17	248424.0	July	2024
2	2024-07-16	249559.0	July	2024
3	2024-07-15	233209.0	July	2024
4	2024-07-14	119283.0	July	2024

```
…          …        …     …    …
1596 2020-03-04  311662.0  March  2020
1597 2020-03-03  319727.0  March  2020
1598 2020-03-02  321569.0  March  2020
1599 2020-03-01       NaN  March  2020
1600 2023-06-19  196645.0   June  2023

[1601 rows x 4 columns]
```

[ ]: