Bagels on Broadway - graphs

November 25, 2024

[1]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

[2]:
```
df = pd.read_csv('Extra_Data_Analysis.csv')
```

[3]:
```
df
```

[3]:

	Postal Code	15 to 19 years - Counts - Total	20 to 24 years - Counts - Total	25 to 29 years - Counts - Total	Between 7 a.m. and 7:59 a.m. - Counts - Total
0	M4N	1125	890	715	695
1	M4P	855	1500	3110	1610
2	M4R	795	665	825	655
3	M4S	1115	1420	3165	1755
4	M4T	455	480	610	425
5	M4V	695	855	1580	1095
6	M5N	1140	980	870	
7	M5P	1025	1060	1585	
8	M5R	745	1760	2940	

6	980	
7	1050	
8	1145	

	Between 8 a.m. and 8:59 a.m. - Counts - Total	Bicycle - Counts - Total	\
0	1115	40	
1	1925	175	
2	785	55	
3	2260	160	
4	600	60	
5	1380	140	
6	1165	70	
7	1365	145	
8	1980	455	

	Population, 2021 - Counts - Total	Public transit - Counts - Total	\
0	16058	705	
1	25057	2560	
2	11909	825	
3	30754	2860	
4	10332	560	
5	19273	1190	
6	16154	1025	
7	19791	1280	
8	26197	1540	

	Total - Main mode of commuting - Counts - Total	…	\
0	3065	…	
1	6735	…	
2	2645	…	
3	7385	…	
4	1920	…	
5	4575	…	
6	4050	…	
7	4865	…	
8	6095	…	

	percentage_leaving_early_morning	percentage_bicycle	percentage_walked	\
0	59.053834	1.305057	9.624796	
1	52.487008	2.598367	12.249443	
2	54.442344	2.079395	10.018904	
3	54.366960	2.166554	11.103588	
4	53.385417	3.125000	13.020833	
5	54.098361	3.060109	17.267760	
6	52.962963	1.728395	7.407407	
7	49.640288	2.980473	9.352518	
8	51.271534	7.465135	23.625923	

```
   percentage_transit  percentage_commuting  percentage_car  number_car  \
0           23.001631             33.931485       66.068516        2025
1           38.010393             52.858203       47.141797        3175
2           31.190926             43.289225       56.710775        1500
3           38.727150             51.997292       48.002708        3545
4           29.166667             45.312500       54.687500        1050
5           26.010929             46.338798       53.661202        2455
6           25.308642             34.444444       65.555556        2655
7           26.310380             38.643371       61.356629        2985
8           25.266612             56.357670       43.642330        2660

   percentage_WFH  percentage_15_to_29  sum_15_to_29
0       54.178886            17.000872          2730
1       51.965994            21.810273          5465
2       56.623586            19.187169          2285
3       56.313794            18.534174          5700
4       60.486322            14.953542          1545
5       54.941860            16.240336          3130
6       46.822309            18.509348          2990
7       52.360097            18.543783          3670
8       55.149982            20.784823          5445

[9 rows x 24 columns]
```

[4]:
```
df.columns
```

[4]:
```
Index(['Postal Code', '15 to 19 years - Counts - Total',
       '20 to 24 years - Counts - Total', '25 to 29 years - Counts - Total',
       'Between 7 a.m. and 7:59 a.m. - Counts - Total',
       'Between 8 a.m. and 8:59 a.m. - Counts - Total',
       'Bicycle - Counts - Total', 'Population, 2021 - Counts - Total',
       'Public transit - Counts - Total',
       'Total - Main mode of commuting - Counts - Total',
       'Total - Place of work status - Counts - Total',
       'Total - Time leaving for work - Counts - Total',
       'Walked - Counts - Total', 'Worked at home - Counts - Total',
       'percentage_leaving_early_morning', 'percentage_bicycle',
       'percentage_walked', 'percentage_transit', 'percentage_commuting',
       'percentage_car', 'number_car', 'percentage_WFH', 'percentage_15_to_29',
       'sum_15_to_29'],
      dtype='object')
```

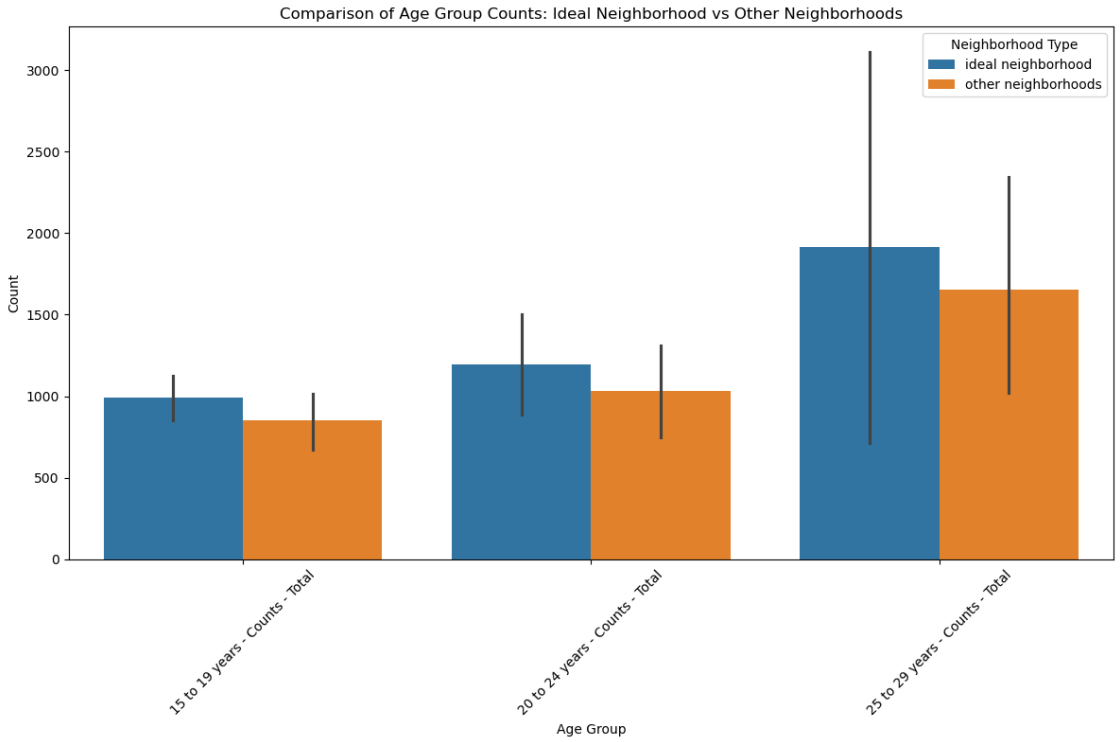
[5]:
```
df.dtypes
```

[5]:
```
Postal Code                                  object
15 to 19 years - Counts - Total               int64
```

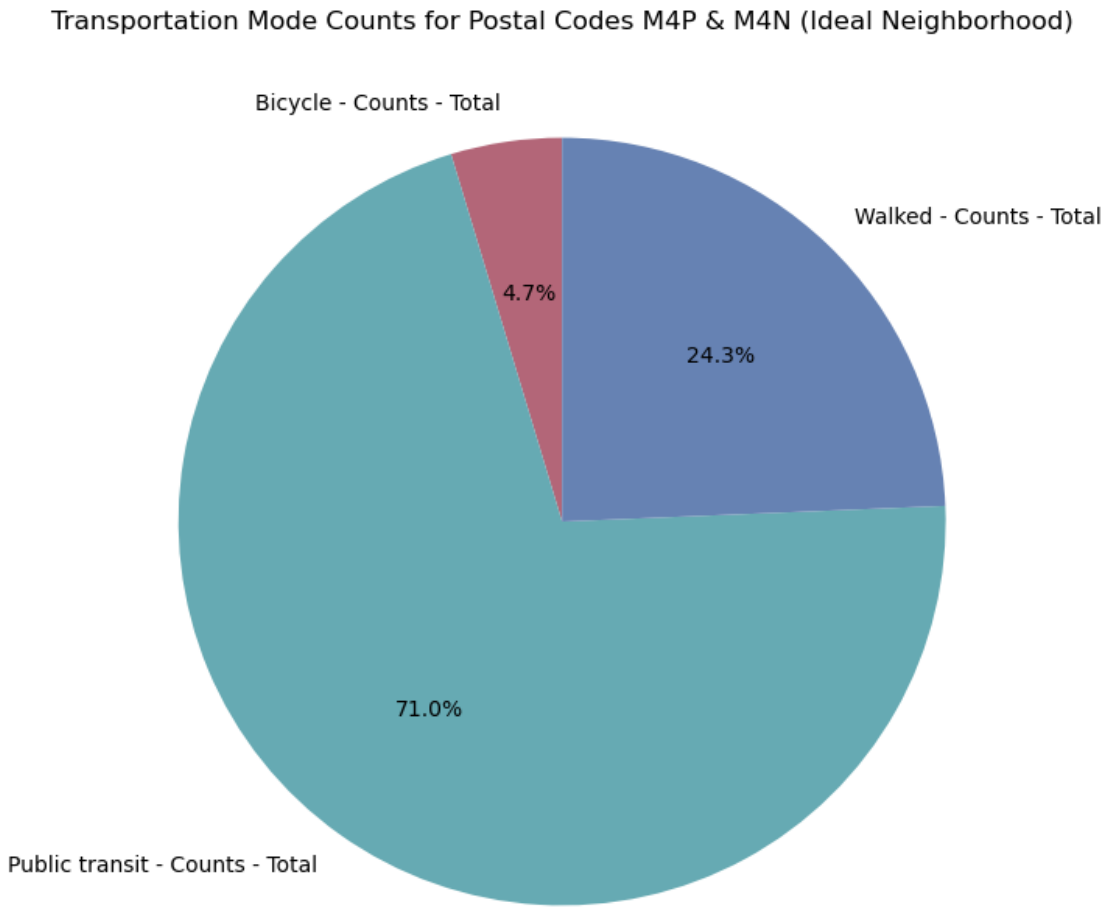
```
20 to 24 years - Counts - Total                          int64
25 to 29 years - Counts - Total                          int64
Between 7 a.m. and 7:59 a.m. - Counts - Total            int64
Between 8 a.m. and 8:59 a.m. - Counts - Total            int64
Bicycle - Counts - Total                                 int64
Population, 2021 - Counts - Total                        int64
Public transit - Counts - Total                          int64
Total - Main mode of commuting - Counts - Total          int64
Total - Place of work status - Counts - Total            int64
Total - Time leaving for work - Counts - Total           int64
Walked - Counts - Total                                  int64
Worked at home - Counts - Total                          int64
percentage_leaving_early_morning                       float64
percentage_bicycle                                     float64
percentage_walked                                      float64
percentage_transit                                     float64
percentage_commuting                                   float64
percentage_car                                         float64
number_car                                               int64
percentage_WFH                                         float64
percentage_15_to_29                                    float64
sum_15_to_29                                             int64
dtype: object
```

[6]:
```
# Define the list of postal codes for the "other neighborhoods"
other_postal_codes = ['M4R', 'M4S', 'M4T', 'M4V', 'M5N', 'M5P', 'M5R']

# Create a new column 'Neighborhood Type'
df['Neighborhood Type'] = df['Postal Code'].apply(
    lambda x: 'ideal neighborhood' if x in ['M4P', 'M4N'] else 'other␣
 ↪neighborhoods'
)

# --- Bar Graph: Age Group Counts (15-19, 20-24, 25-29) Comparison ---
# Select the relevant age group columns for comparison
age_group_columns = ['15 to 19 years - Counts - Total',
                     '20 to 24 years - Counts - Total',
                     '25 to 29 years - Counts - Total']

# Filter the data for both the neighborhood types
ideal_neighborhood_df = df[df['Neighborhood Type'] == 'ideal neighborhood']
other_neighborhood_df = df[df['Neighborhood Type'] == 'other neighborhoods']

# Melt the data for easy plotting (reshape from wide to long format)
melted_ideal_df = ideal_neighborhood_df.melt(id_vars=['Neighborhood Type'],␣
 ↪value_vars=age_group_columns,
```

```
                                                    var_name='Age Group',␣
 ↪value_name='Count')
melted_other_df = other_neighborhood_df.melt(id_vars=['Neighborhood Type'],␣
 ↪value_vars=age_group_columns,
                                                    var_name='Age Group',␣
 ↪value_name='Count')

# Combine both dataframes to plot together
combined_df = pd.concat([melted_ideal_df, melted_other_df])

# Plotting the bar graph for "ideal neighborhood" vs "other neighborhoods"
plt.figure(figsize=(12, 8))
sns.barplot(x='Age Group', y='Count', hue='Neighborhood Type', data=combined_df)

# Customizing the plot for age groups
plt.title('Comparison of Age Group Counts: Ideal Neighborhood vs Other␣
 ↪Neighborhoods')
plt.xlabel('Age Group')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.tight_layout()
plt.legend(title='Neighborhood Type')

# Show the bar plot
plt.show()
```



[7]:
```
# Create a new column 'Neighborhood Type' based on the 'Postal Code'
df['Neighborhood Type'] = df['Postal Code'].apply(
    lambda x: 'ideal neighborhood' if x in ['M4P', 'M4N'] else 'other␣
 ↪neighborhoods'
)

# Filter DataFrame for Postal Code M4P & M4N (Ideal Neighborhood)
filtered_df = df[df['Neighborhood Type'] == 'ideal neighborhood']

# Sum the counts for each of the columns of interest for the filtered postal␣
 ↪codes
transport_columns = ['Bicycle - Counts - Total', 'Public transit - Counts -␣
 ↪Total', 'Walked - Counts - Total']
sum_counts = filtered_df[transport_columns].sum()

# Plotting the pie chart with updated colors (Green, Pink, Blue)
plt.figure(figsize=(8, 8))
sum_counts.plot(kind='pie', autopct='%1.1f%%', startangle=90,␣
 ↪colors=['#b36678', '#66aab3', '#6682b3'], legend=False)

# Adding title and labels
plt.title('Transportation Mode Counts for Postal Codes M4P & M4N (Ideal␣
 ↪Neighborhood)')
plt.ylabel('')  # Hide the ylabel for better presentation

# Show the plot
plt.show()
```



[8]:
```
pf = pd.read_csv('Available income per capita in Ontario 1981-2022(Sheet1).csv')
```

[9]:
```
pf
```

[9]:

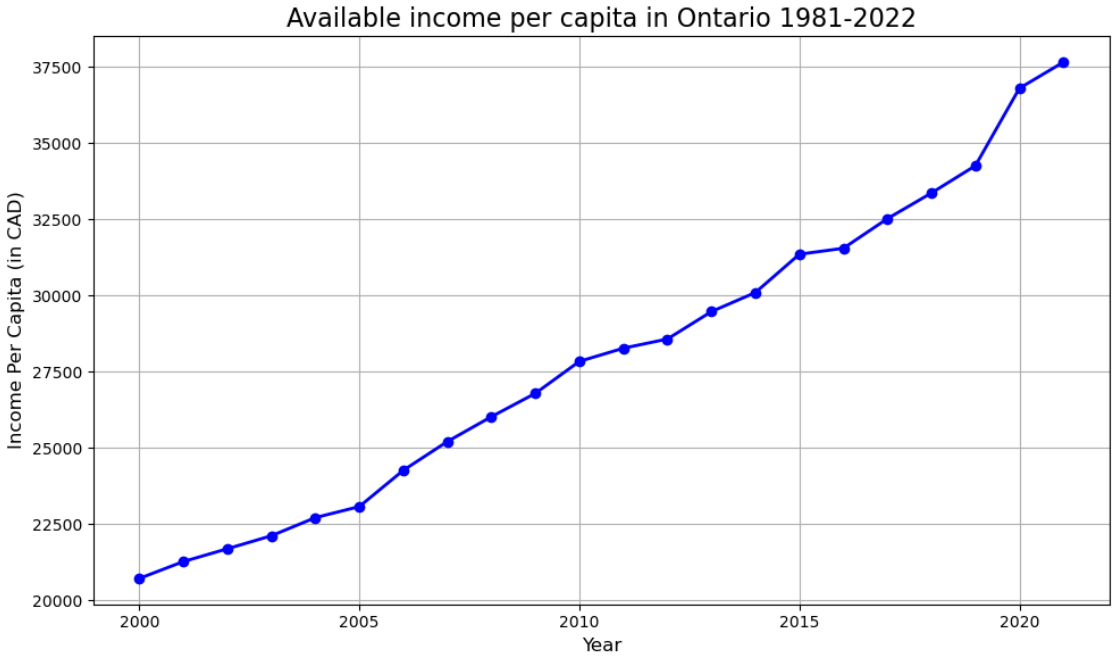
	Year	Ontario
0	2000	20724.0
1	2001	21277.0
2	2002	21697.0
3	2003	22123.0
4	2004	22718.0
5	2005	23081.0
6	2006	24270.0
7	2007	25213.0
8	2008	26027.0
9	2009	26793.0
10	2010	27841.0

```
11  2011  28273.0
12  2012  28574.0
13  2013  29473.0
14  2014  30099.0
15  2015  31355.0
16  2016  31550.0
17  2017  32521.0
18  2018  33362.0
19  2019  34259.0
20  2020  36796.0
21  2021  37653.0
```

[10]:
```
pf.dtypes
```

[10]:
```
Year         int64
Ontario    float64
dtype: object
```

[11]:
```
# Plotting the line chart
plt.figure(figsize=(10, 6))

# Plot the data with 'Year' on x-axis and 'Ontario' on y-axis
plt.plot(pf['Year'], pf['Ontario'], marker='o', color='b', linestyle='-',␣
 ↪linewidth=2)

# Adding title and labels
plt.title('Available income per capita in Ontario 1981-2022', fontsize=16)
plt.xlabel('Year', fontsize=12)
plt.ylabel('Income Per Capita (in CAD)', fontsize=12)

# Displaying the plot
plt.grid(True)
plt.tight_layout()
plt.show()
```


Available income per capita in Ontario 1981-2022
Income Per Capita (in CAD)
Year
37500
35000
32500
30000
27500
25000
22500
20000
2000
2005
2010
2015
2020