Spatial Analysis of Demographic, Housing Prices and Public Transportation Accessibility in Chicago

Yizhou Wang & Hongtao Huang

Master of Regional Planning 25'

1 Introduction

Finding an affordable home with easy access to public transportation is a challenge in many cities. Therefore, the relationship between housing prices and public transportation accessibility has become a topic of significant interest, particularly in densely populated urban areas. This research aims to investigate the correlation between median housing values and the availability of public transit routes and stops in Chicago, during the year 2021 and 2018. By examining this relationship, the study seeks to provide valuable insights into the complex interplay between transportation infrastructure, housing affordability, and urban development in a densely populated urban area with intense commute in the Chicago metropolitan area. The findings of this research could inform policymakers, urban planners, and stakeholders in their efforts to promote sustainable and equitable urban growth while ensuring accessibility to essential services.

2 Problem Statement

This research question addresses the gap in knowledge regarding how public transit accessibility and ridership variations within Chicago, influence housing prices, and also the demographic and economic properties. Existing research often focuses on broader metropolitan areas or the gentrification effects of new transit lines, however, a more detailed effect of how transit stations influencing housing price in commuter towns are rare. This study will analyze data from 2021, 2018 to determine if housing prices correlate with proximity to different public transportation options (subway stations, bus stops); ridership of the stations nearby, and some demographic and economic indexes across Chicago's diverse communities. Moreover, it is also examining the difference of ridership of Covid and Post-Covid era.

It is hypothesize that in Chicago, house prices are positively correlated by the density of the spatial distribution of public transportation, while at the same time, house prices determine to some extent the housing vacancy in the region as well as the population race, nationality and per capita median income etc.

3 Literature Review

3.1 Housing Prices and Public Transportation

Recent research has highlighted the intricate relationship between housing prices and public transportation accessibility. Studies by Hess and Almeida (2007) and Lee and Smart (2020) have demonstrated that residential property values tend to increase as the distance to public transit stations decreases, reflecting the importance of transportation convenience for urban residents. Palma (2019) stated that:

"10 minutes less travel time to work imply a 2.8% increase in housing price. The price is very sensitive to socio-economic structure of the commune: a 10% increase in the proportion of one member households causes a 50% increase of the price."

However, this positive correlation has also raised concerns about potential negative impacts, such as the displacement of low-income residents due to rising housing costs. Lee and Smart's (2020) findings suggest that areas with high transportation convenience, particularly those with direct access to major employment centers like New York City, may experience significant increases in poverty rates and the migration of low-income groups to these densely populated areas.

However, there are contraditory finding from the research of Brueckner and Rosenthal (2009). They indicates that communities with concentrated public transportation are more likely to have higher populations of low-income families, as public transit offers a more affordable alternative to car ownership. Conversely, Giulano (2005) found that wealthier residents tend to avoid areas with predominantly bus-based transit systems due to longer travel times.

The Hedonic Price Model is particularly useful in the real estate market, especially in analyzing the relationship between housing prices and public transportation convenience. By decomposing housing prices into contributions of different characteristics, this model can quantitatively evaluate the impact of public transportation proximity on housing value, such as showing that houses near subway stations may be a certain percentage higher than the market average price. This analysis is crucial for urban planners and policymakers as it can guide them to increase housing supply or improve public transportation services in areas with convenient transportation. In addition, the results of the consumer price model can also help real estate developers and investors make wiser investment decisions and predict the potential impact of public transportation development on housing prices (Herath & Maier,2010).

3.2 Public Transportation in Chicago

Chicago Department of Transportation The Chicago Department of Transportation is a major participant in the Regional Transportation Authority (RTA). The Regional Transportation Authority oversees the operations of the Chicago Transit Authority (CTA), the Metra rail system, and the Pace bus system. Crum (2020) believes that the establishment of new CTA stations will cause the value of surrounding houses to increase, which promotes gentrification.

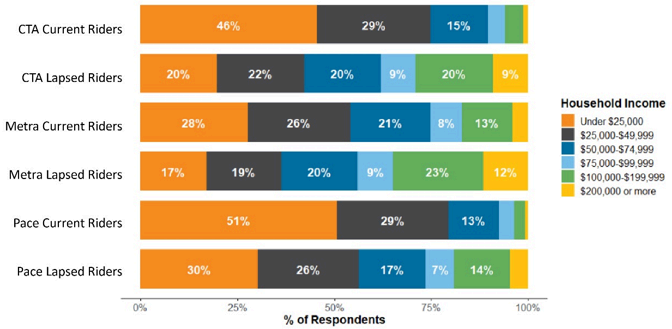
Moreover, previous studies have examined Chicago and its surrounding areas, providing a valuable research context for exploring this topic. As a densely populated region with a well-established public transportation network and diverse commuter communities, it offers an opportunity to examine the interplay between housing prices, transit accessibility, and socioeconomic factors.

Some of the major cities have substantially greater percentages of workers reliant on transportation than the national average (5.0%), such as Chicago (28.3%) (U.S. Census Bureau, 2018). With a significant portion of Chicago residents relying on public transportation, the area serves as an ideal case study for understanding the complex dynamics between transportation development, housing affordability, and sustainable urban growth.

3.2.1 The Demographic of Public Transportation in Chicago

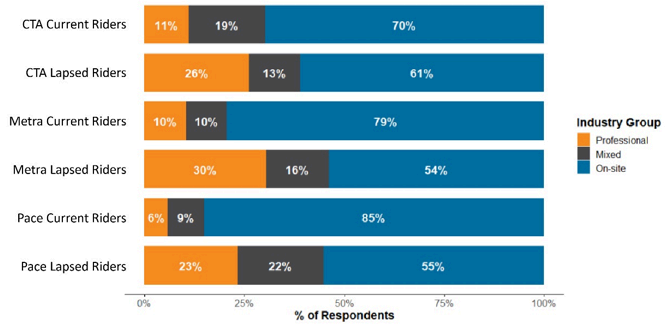
Chicago also has experienced significant gentrification in its poor neighborhoods, with both housing affordability and transit affordability potentially declining for the minority residents who tend to be low-income (Liu et al., 2021). According to a survey by the Regional Transportation Authority (2021), the majority of current users believe transportation is safe and are either vital workers or minorities. Here are some figures from the survey. It could be informed that each the users of the different transportation means, CTA, Metra, and Pace has slightly different patterns of household income and racial demographics.

Income by Service Board and Rider Type



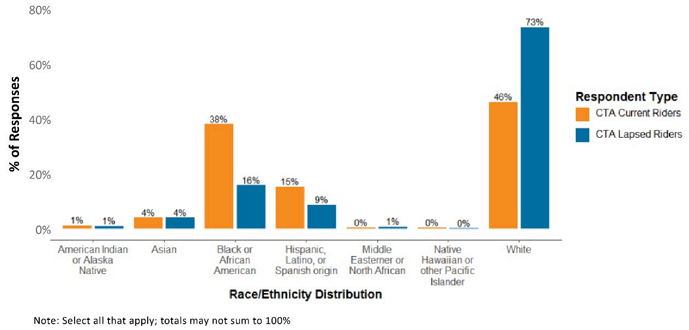
Income by Service Board and Rider Type

Industry Group by Service Board and Rider Type



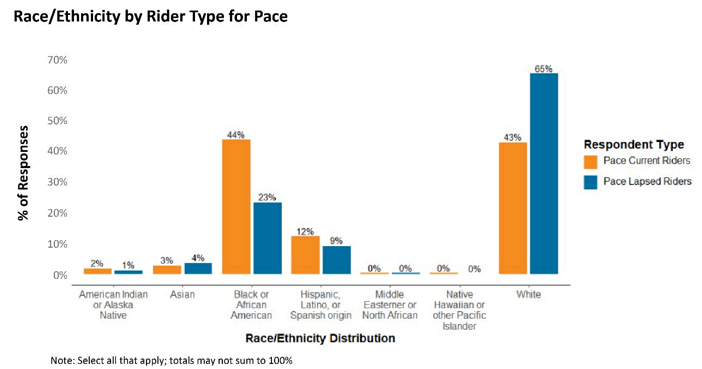
Industry Group by Service Board and Rider Type

Race/Ethnicity by Rider Type for CTA



Note: Select all that apply; totals may not sum to 100%

Race/Ethnicity by Rider Type for CTA



Race/Ethnicity by Rider Type for Pace

3.2.2 Influence of Covid to the Public Transit in Chicago

According to TransitCenter(2021), during the early stages of the pandemic, there was a sharp decline in ridership as lockdowns were implemented and many people avoided public spaces to reduce the risk of transmission. This decline forced transit agencies to reduce services due to decreased demand and financial strain from lost fare revenue. In some areas, service cuts have made it difficult for essential workers—who are disproportionately people of color—to commute, as they are more likely to rely on public transport. In 2021, 10% of Black workers, 5% of Latinx workers, and 5% of Asian workers in the Chicago Metropolitan Statistical Area (compared to 4% of White workers) rode buses to work. Since the COVID-19 pandemic began, people of color are more likely than white people to continue commuting to and from get off work. In 2021, 18% of Black workers, 11% of Latinx workers, 30% of Asian workers and 25% of white workers living in the Chicago area worked remotely.

Therefore, Sukaryavichute and Prytherch (2018) argues that transportation systems are a critical component in achieving economic and social well-being, especially in the context of social justice. Fair transportation discussions should include reflection and improvement on policies to ensure equitable access to transportation services for all, especially disadvantaged groups. This involves critical analysis of existing inequalities and ensuring that transport systems are designed to meet the needs of all, especially those with lower incomes who rely on public transport. In Chicago, studying the connection between housing affordability and public transportation is particularly important because it can reveal and address social inequalities in urban planning. By understanding the interaction between the two, we can more effectively promote equitable urban development strategies, improve the quality of life of low-income residents, and promote the balanced development of society as a whole.

4 Methodology plan

4.1 Data:

4.1.1 Demographic and Economic data in Block Group: Census API

Housing Data:

1. `B25077_001E` : Median value (dollars) of owner-occupied housing units.
2. `B25085_001E` : Price asked for vacant-for-sale-only and sold, not occupied housing units.
3. `B25001_001E` : Total housing units.
4. `B25002_003E` : Vacant housing units.
5. `B25003_002E` : Owner-occupied housing units.
6. `B25003_003E` : Renter-occupied housing units.

Demographic Data:

1. `B06011_001E` : Median income in the past 12 months (in 2019 inflation-adjusted dollars) by place of birth in the United States.
2. `B05007_001E` : Foreign-born population.
3. `B02001_001E` : Total population by race
4. `B02001_002E` : Total white population by race

4.1.2 Public Transit Data in Chicago

Stops and Routes in Chicago is available from the datahub of the Chicago Metropolitan Agency for Planning (CMAP)

https://datahub.cmap.illinois.gov/search?categories=%252Fcategories%252Ftransit

Ridership for different Routes is available from the Regional Transportation Authority (RTA) - Mapping and Statistics

https://rtams.org/ridership

Stops:

- 2023 CTA Bus Stops - February 7, 2023
- 2023 Pace Bus Stops - May 1, 2023
- 2023 Metra Rail Stations - 2023

Routes/Lines:

- 2023 Pace Bus Routes - 2023
- 2023 CTA Bus Routes - February 7, 2023
- 2023 CTA Rail Lines - February 7, 2023
- 2023 Metra Rail Lines - 2023

Ridership:

- Metra Station Ridership (Boarding & Alighting Survey) 2002-2018
- Metra Total Rail Ridership By Line 2015-2023
- CTA Rail Ridership By Station 2015-2023
- CTA Bus Ridership By Route 2015-2023
- Pace Bus Ridership by Route 2015-2023 p by Route 2015-2023

4.2 Analysis methods

The unit of analysis is the 866 census tracts within the county. The accessibility to public transport is indicated by the density of public transportation stops.

For this study, Global and Local Moran's I is used to analyze the basic data pattern along with the simple visualizations. Local Global Moran's I is used to evaluate the spatial autocorrelation of housing prices. Global Moran's I is used to measure spatial autocorrelation, determining whether similar values of a variable are clustered together geographically, which helps in identifying patterns such as the clustering or dispersion of demographic groups, real estate values, or accessibility features across a given area.

OLS, spatial regime, spatial lag model, and Spatial Durbin Model are applied to figure out what indicators are related to house prices. For this topic, it is possible to explore the relationship between housing prices and public transportation stops, the corresponding ridership, the distance between each tract and the closest stop, the percentage of non-whites, the median per capita income, and the percentage of immigrants and build a best-fit regression model around housing prices. This analysis will help identify the most significant predictors of housing prices in the city of Chicago and understand the impact of public transportation accessibility on these prices, controlling for other factors.

5 Site Analysis of Basic Data

5.1 Census Demographic Data

5.1 Data Processing

In [1]:
```
import pandas as pd
import numpy as np
import geopandas as gpd
import matplotlib.pyplot as plt
import seaborn as sns
import pysal as py
from matplotlib import colors
from pysal.viz import splot
from esda.moran import Moran_Local
from splot.esda import plot_moran
sns.set_context(context='paper')
%matplotlib inline
from census import Census
from us import states
from shapely.geometry import Point
from pyproj import Proj, transform
from geopy.distance import great_circle
from pysal.explore import esda
from pysal.lib import weights
from numpy.random import seed
import rioxarray  # Surface data manipulation
import xarray  # Surface data manipulation
from pysal.model import spreg
import statsmodels.formula.api as sm
## warnings is a module that allows you to filter warnings
import warnings
## we are going to ignore all warnings (so they won't print)
warnings.filterwarnings("ignore")
```

```
C:\Users\86158\anaconda3\envs\gds_py\lib\site-packages\numba\core\decorators.py:262: NumbaDeprecationWarning: numba.generated_jit is deprecated. Please see the documentation at: https://numba.readthedocs.io/en/stable/reference/deprecation.html#deprecation-of-generated-jit for more information and advice on a suitable replacement.
  warnings.warn(msg, NumbaDeprecationWarning)
C:\Users\86158\anaconda3\envs\gds_py\lib\site-packages\quantecon\lss.py:20: NumbaDeprecationWarning: The 'nopython' keyword argument was not supplied to the 'numba.jit' decorator. The implicit default value for this argument is currently False, but it will be changed to True in Numba 0.59.0. See https://numba.readthedocs.io/en/stable/reference/deprecation.html#deprecation-of-object-mode-fall-back-behaviour-when-using-jit for details.
  def simulate_linear_model(A, x0, v, ts_length):
C:\Users\86158\anaconda3\envs\gds_py\lib\site-packages\spaghetti\network.py:40: FutureWarning: The next major release of pysal/spaghetti (2.0.0) will drop support for all ``libpysal.cg`` geometries. This change is a first step in refactoring ``spaghetti`` that is expected to result in dramatically reduced runtimes for network instantiation and operations. Users currently requiring network and point pattern input as ``libpysal.cg`` geometries should prepare for this simply by converting to ``shapely`` geometries.
  warnings.warn(dep_msg, FutureWarning, stacklevel=1)
```

In [2]:
```
c = Census("d9c002dc1334c8f6cbea48d3f10a4176cdf89064")
```

In [3]:
```
# Retrieving census data for Cook County, Illinois, for the year 2021
_2021_census = c.acs5.state_county_tract(fields=('NAME', 'B25077_001E', 'B25085_001E', 'B25001_001E', 'B25002_003E', 'B25003_002E', 'B25003_003E', 'B06011_001E', 'B05007_001E', 'B02001_001E','B02001_002E'),
                                         state_fips=states.IL.fips,
                                         county_fips="031",  # Cook County, Illinois
                                         tract="*",
                                         year=2021)
# Retrieving census data for Cook County, Illinois, for the year 2021
_2018_census = c.acs5.state_county_tract(fields=('NAME', 'B25077_001E', 'B25085_001E', 'B25001_001E', 'B25002_003E', 'B25003_002E', 'B25003_003E', 'B06011_001E', 'B05007_001E', 'B02001_001E','B02001_002E'),
                                         state_fips=states.IL.fips,
                                         county_fips="031",  # Cook County, Illinois
                                         tract="*",
                                         year=2018)
```

In [4]:
```
#_2021_census
```

In [5]:
```
# Convert the data to a pandas dataframe
acs_df_2021 = pd.DataFrame(_2021_census)
# Create a GEOID column by concatenating the state, county, and tract codes
acs_df_2021["GEOID"] = acs_df_2021["state"] + acs_df_2021["county"] + acs_df_2021["tract"]
# Calculate the percentage of non-White population by subtracting the White population from the total population and dividing by the total population
acs_df_2021["nonwhite_perc"] = (acs_df_2021["B02001_001E"] - acs_df_2021["B02001_002E"]) / acs_df_2021["B02001_001E"]
acs_df_2021["foreign_perc"] = acs_df_2021["B05007_001E"] / acs_df_2021["B02001_001E"]
```

In [6]:
```
acs_df_2021["vacant_unit_perc"] = acs_df_2021["B25002_003E"] / acs_df_2021["B25001_001E"]
acs_df_2021["owner_occupied_perc"] = acs_df_2021["B25003_002E"] / acs_df_2021["B25001_001E"]
acs_df_2021["renter_occupied_perc"] = acs_df_2021["B25003_003E"] / acs_df_2021["B25001_001E"]
```

In [7]:
```
# Convert the data to a pandas dataframe
acs_df_2018 = pd.DataFrame(_2018_census)
# Create a GEOID column by concatenating the state, county, and tract codes
acs_df_2018["GEOID"] = acs_df_2018["state"] + acs_df_2018["county"] + acs_df_2018["tract"]
# Calculate the percentage of non-White population by subtracting the White population from the total population and dividing by the total population
acs_df_2018["nonwhite_perc"] = (acs_df_2018["B02001_001E"] - acs_df_2018["B02001_002E"]) / acs_df_2018["B02001_001E"]
acs_df_2018["foreign_perc"] = acs_df_2018["B05007_001E"] / acs_df_2018["B02001_001E"]
```

In [8]:
```
acs_df_2018["vacant_unit_perc"] = acs_df_2018["B25002_003E"] / acs_df_2018["B25001_001E"]
acs_df_2018["owner_occupied_perc"] = acs_df_2018["B25003_002E"] / acs_df_2018["B25001_001E"]
acs_df_2018["renter_occupied_perc"] = acs_df_2018["B25003_003E"] / acs_df_2018["B25001_001E"]
```

In [9]:
```
# Read the shapefile for Cook County
cook_tract = gpd.read_file("https://www2.census.gov/geo/tiger/TIGER2021/TRACT/tl_2021_17_tract.zip")
# Filter the tracts for Cook County only
cook_tract = cook_tract[cook_tract["COUNTYFP"] == "031"]
```

In [10]:
```
cook_census_geo_2021 = cook_tract.merge(acs_df_2021, left_on = 'GEOID', right_on = 'GEOID')
cook_census_geo_2021.describe()
```

Out[10]:

	ALAND	AWATER	B25077_001E	B25085_001E	B25001_001E	B25002_003E	B25003_002E	B25003_003E	B06011_001E	B05007_001E	B02001_001E	B02001_002E	nonwhite_perc	foreign_perc	vacant_unit_perc	owner_occupied_perc	renter_occupied_perc
count	1.332000e+03	1.332000e+03	1.332000e+03	1332.000000	1332.000000	1332.000000	1332.000000	1332.000000	1.332000e+03	1332.000000	1332.000000	1332.000000	1328.000000	1328.000000	1328.000000	1328.000000	1328.000000
mean	1.837350e+06	1.341090e+06	-9.719136e+06	20.870120	1692.144144	157.115616	882.239489	652.789039	-3.463530e+06	827.337087	3953.001502	2057.081081	0.511296	0.192016	0.097898	0.515126	0.386976
std	3.147841e+06	4.704685e+07	8.114146e+07	28.637381	835.675693	138.700517	564.369966	533.515014	4.822262e+07	778.174197	1778.569024	1577.538675	0.294288	0.142000	0.071966	0.233497	0.201916
min	0.000000e+00	0.000000e+00	-6.666667e+08	0.000000	0.000000	0.000000	0.000000	0.000000	-6.666667e+08	0.000000	0.000000	0.000000	0.034380	0.000000	0.000000	0.000000	0.000000
25%	4.110900e+05	0.000000e+00	1.736750e+05	0.000000	1124.750000	70.000000	423.500000	295.000000	2.683000e+04	210.000000	2560.250000	656.000000	0.255031	0.071450	0.048706	0.326816	0.220682
50%	8.969725e+05	0.000000e+00	2.483000e+05	10.000000	1614.500000	122.000000	794.000000	526.000000	3.472350e+04	589.000000	3885.000000	1895.500000	0.453815	0.165461	0.081027	0.500264	0.386741
75%	2.147472e+06	0.000000e+00	3.627000e+05	33.000000	2114.000000	203.250000	1264.500000	833.000000	4.929225e+04	1256.000000	5145.750000	3170.250000	0.777141	0.300599	0.125492	0.711206	0.541907
max	6.184753e+07	1.717072e+09	1.474100e+06	219.000000	7694.000000	1345.000000	3108.000000	4750.000000	1.147500e+05	4543.000000	11295.000000	8515.000000	1.000000	0.669796	0.477829	1.000000	0.985468

In [11]:
```
cook_census_geo_2018 = cook_tract.merge(acs_df_2018, left_on = 'GEOID', right_on = 'GEOID')
cook_census_geo_2018.describe()
```

Out[11]:

	ALAND	AWATER	B25077_001E	B25085_001E	B25001_001E	B25002_003E	B25003_002E	B25003_003E	B06011_001E	B05007_001E	B02001_001E	B02001_002E	nonwhite_perc	foreign_perc	vacant_unit_perc	owner_occupied_perc	renter_occupied_perc
count	1.288000e+03	1.288000e+03	1.288000e+03	1288.000000	1288.000000	1288.000000	1288.000000	1288.000000	1.288000e+03	1288.000000	1288.000000	1288.000000	1284.000000	1284.000000	1284.000000	1284.000000	1284.000000
mean	1.742769e+06	1.382477e+06	-7.502018e+06	28.131988	1635.746118	169.021739	829.319099	637.405280	-2.036330e+06	821.369565	3914.440217	2212.215839	0.469626	0.191050	0.109513	0.496794	0.393693
std	2.648591e+06	4.784372e+07	7.158056e+07	33.609735	771.371540	138.654729	541.363786	478.766688	3.711046e+07	789.686418	1732.385134	1663.625347	0.314474	0.145032	0.078187	0.230792	0.194565
min	0.000000e+00	0.000000e+00	-6.666667e+08	0.000000	0.000000	0.000000	0.000000	0.000000	-6.666667e+08	0.000000	0.000000	0.000000	0.009518	0.000000	0.000000	0.000000	0.008403
25%	4.019952e+05	0.000000e+00	1.533000e+05	0.000000	1106.000000	76.000000	386.000000	306.000000	2.235900e+04	201.000000	2546.250000	685.750000	0.188690	0.067770	0.055406	0.314252	0.235574
50%	8.629060e+05	0.000000e+00	2.215500e+05	19.000000	1576.000000	136.000000	750.500000	525.500000	3.000000e+04	569.500000	3833.000000	2140.500000	0.398953	0.162402	0.090442	0.474632	0.400432
75%	2.038647e+06	0.000000e+00	3.323750e+05	43.000000	2058.000000	217.250000	1183.000000	829.500000	4.129100e+04	1266.000000	5138.250000	3397.500000	0.758785	0.305400	0.140072	0.687850	0.546442
max	4.514031e+07	1.717072e+09	1.528600e+06	311.000000	6514.000000	1140.000000	3382.000000	3686.000000	1.057860e+05	4893.000000	10317.000000	8467.000000	1.000000	0.691883	0.569182	0.971333	0.942693

In [12]:
```
cook_census_geo_2018.head()
```

Out[12]:

	STATEFP	COUNTYFP	TRACTCE	GEOID	NAME_x	NAMELSAD	MTFCC	FUNCSTAT	ALAND	AWATER	...	B02001_001E	B02001_002E	state	county	tract	nonwhite_perc	foreign_perc	vacant_unit_perc	owner_occupied_perc	renter_occupied_perc
0	17	031	221000	17031221000	2210	Census Tract 2210	G5020	S	220053	0	...	2774.0	2099.0	17	031	221000	0.243331	0.169430	0.089847	0.297394	0.612758
1	17	031	221100	17031221100	2211	Census Tract 2211	G5020	S	440042	0	...	5207.0	3900.0	17	031	221100	0.251008	0.288650	0.072526	0.286263	0.641210
2	17	031	242800	17031242800	2428	Census Tract 2428	G5020	S	439630	0	...	1542.0	1461.0	17	031	242800	0.052529	0.127756	0.079812	0.593114	0.327074
3	17	031	242900	17031242900	2429	Census Tract 2429	G5020	S	323819	0	...	1617.0	1229.0	17	031	242900	0.239951	0.136054	0.107987	0.388076	0.503937
4	17	031	243000	17031243000	2430	Census Tract 2430	G5020	S	324547	0	...	2274.0	1706.0	17	031	243000	0.249780	0.174582	0.040075	0.363467	0.596459

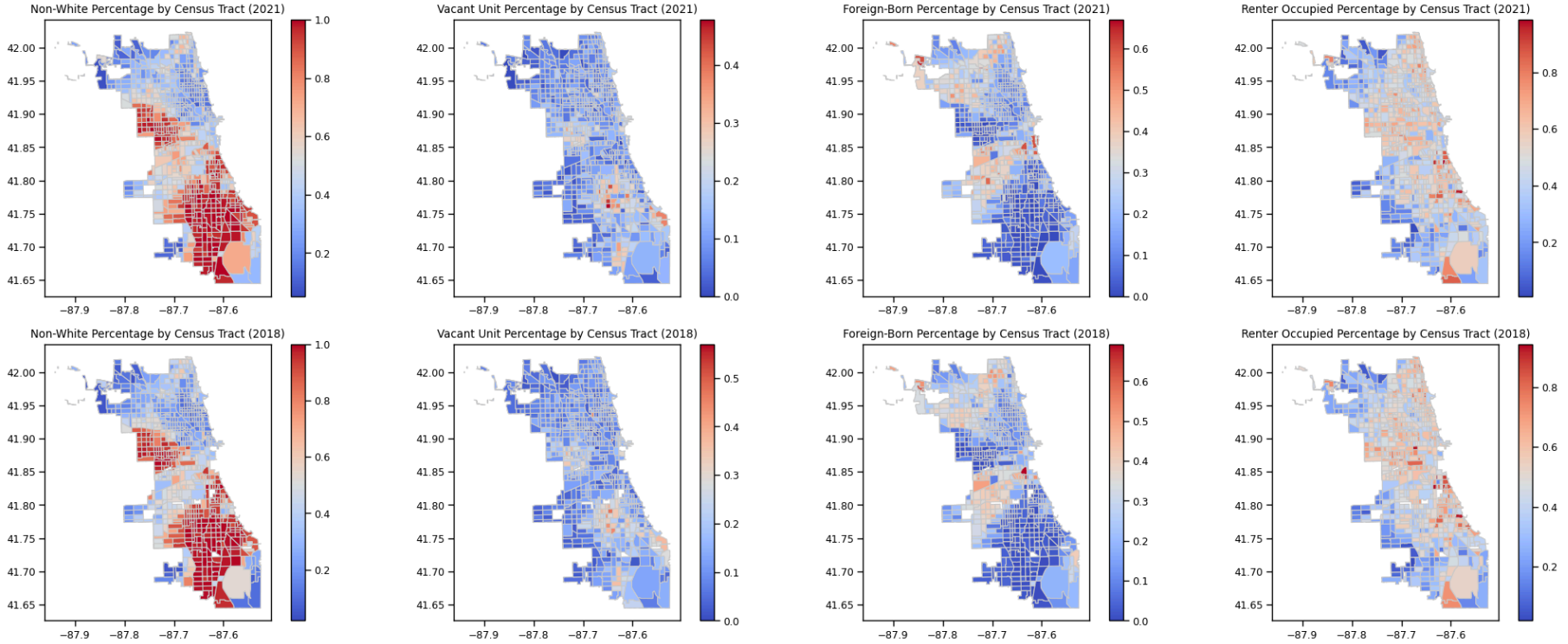
5 rows × 32 columns

In [13]:
```
file_path = "data/Boundaries - Neighborhoods.geojson"
chicago_boundary = gpd.read_file(file_path)
chicago_boundary = chicago_boundary.to_crs(cook_census_geo_2021.crs)
cook_census_geo_2021_clipped = gpd.clip(cook_census_geo_2021, chicago_boundary)
cook_census_geo_2018_clipped = gpd.clip(cook_census_geo_2018, chicago_boundary)
```

5.2 Visualization of Social-Economic Factors and Housing Price in Chicago

5.2.1 Social-Economic Pattern in Chicago

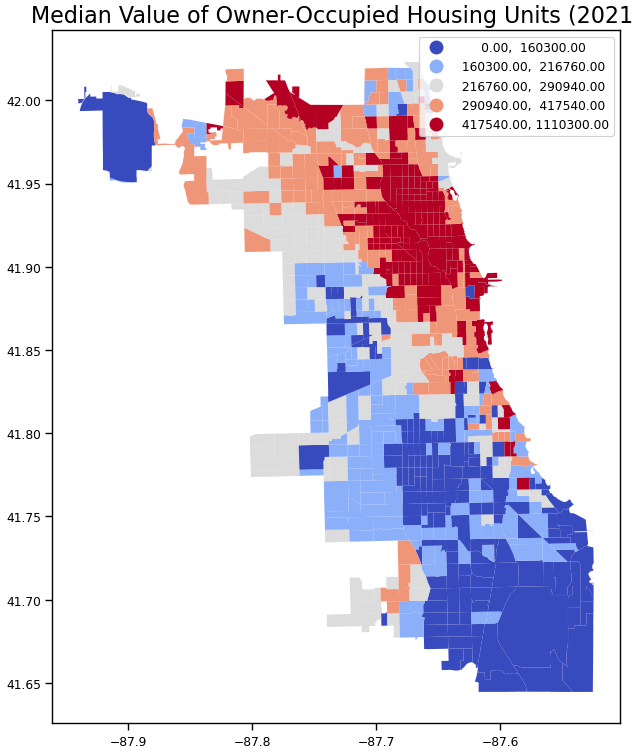
In [96]:
```
# Create a figure and axis object
fig, axs = plt.subplots(2, 4, figsize=(20, 8))
# Plot 2021 data
columns_2021 = ['nonwhite_perc', 'vacant_unit_perc', 'foreign_perc', 'renter_occupied_perc']
titles_2021 = ['Non-White Percentage by Census Tract (2021)', 'Vacant Unit Percentage by Census Tract (2021)', 'Foreign-Born Percentage by Census Tract (2021)', 'Renter Occupied Percentage by Census Tract (2021)']
for i in range(len(columns_2021)):
    cook_census_geo_2021_clipped.plot(column=columns_2021[i], cmap='coolwarm', linewidth=0.8, ax=axs[0, i], edgecolor='0.8', legend=True)
    axs[0, i].set_title(titles_2021[i])
# Plot 2018 data
columns_2018 = ['nonwhite_perc', 'vacant_unit_perc', 'foreign_perc', 'renter_occupied_perc']
titles_2018 = ['Non-White Percentage by Census Tract (2018)', 'Vacant Unit Percentage by Census Tract (2018)', 'Foreign-Born Percentage by Census Tract (2018)', 'Renter Occupied Percentage by Census Tract (2018)']
for i in range(len(columns_2018)):
    cook_census_geo_2018_clipped.plot(column=columns_2018[i], cmap='coolwarm', linewidth=0.8, ax=axs[1, i], edgecolor='0.8', legend=True)
    axs[1, i].set_title(titles_2018[i])
plt.tight_layout()
plt.show()
```

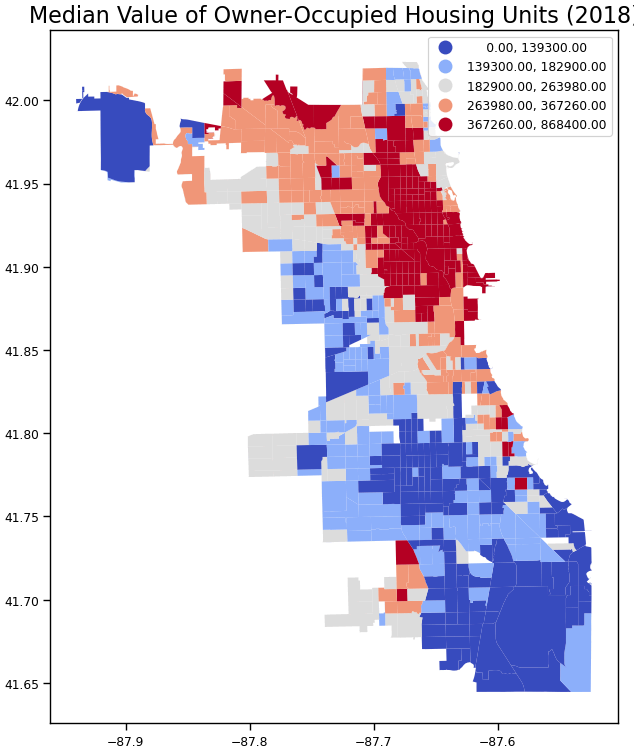


The proportion of non-white residents and the vacancy rate are both higher in the southern part of Chicago, while the foreign population is concentrated in the north and central areas. Most of the housing in the downtown area is rental, whereas the suburbs are predominantly owner-occupied. From 2018 to 2021, the area southwest of the city center showed significant changes in population and housing market, particularly in the proportion of non-white population and vacant housing. This reflects the dynamic development of these regions in terms of increased racial diversity and changes in housing demand.

5.2.1 Housing Price Pattern in Chicago

In [19]:
```
# Create a figure and axis object
fig, ax = plt.subplots(1, 2, figsize=(18, 9))
# Plot the Choropleth map with the filtered data for 2021
cook_census_geo_2021_filtered_clipped.plot(column='B25077_001E', ax=ax[0], legend=True, scheme='Quantiles', cmap='coolwarm')
ax[0].set_title('Median Value of Owner-Occupied Housing Units (2021)', fontsize=16)
# Plot the Choropleth map with the filtered data for 2018
cook_census_geo_2018_filtered_clipped.plot(column='B25077_001E', ax=ax[1], legend=True, scheme='Quantiles', cmap='coolwarm')
ax[1].set_title('Median Value of Owner-Occupied Housing Units (2018)', fontsize=16)
# Show the plot
plt.show()
```

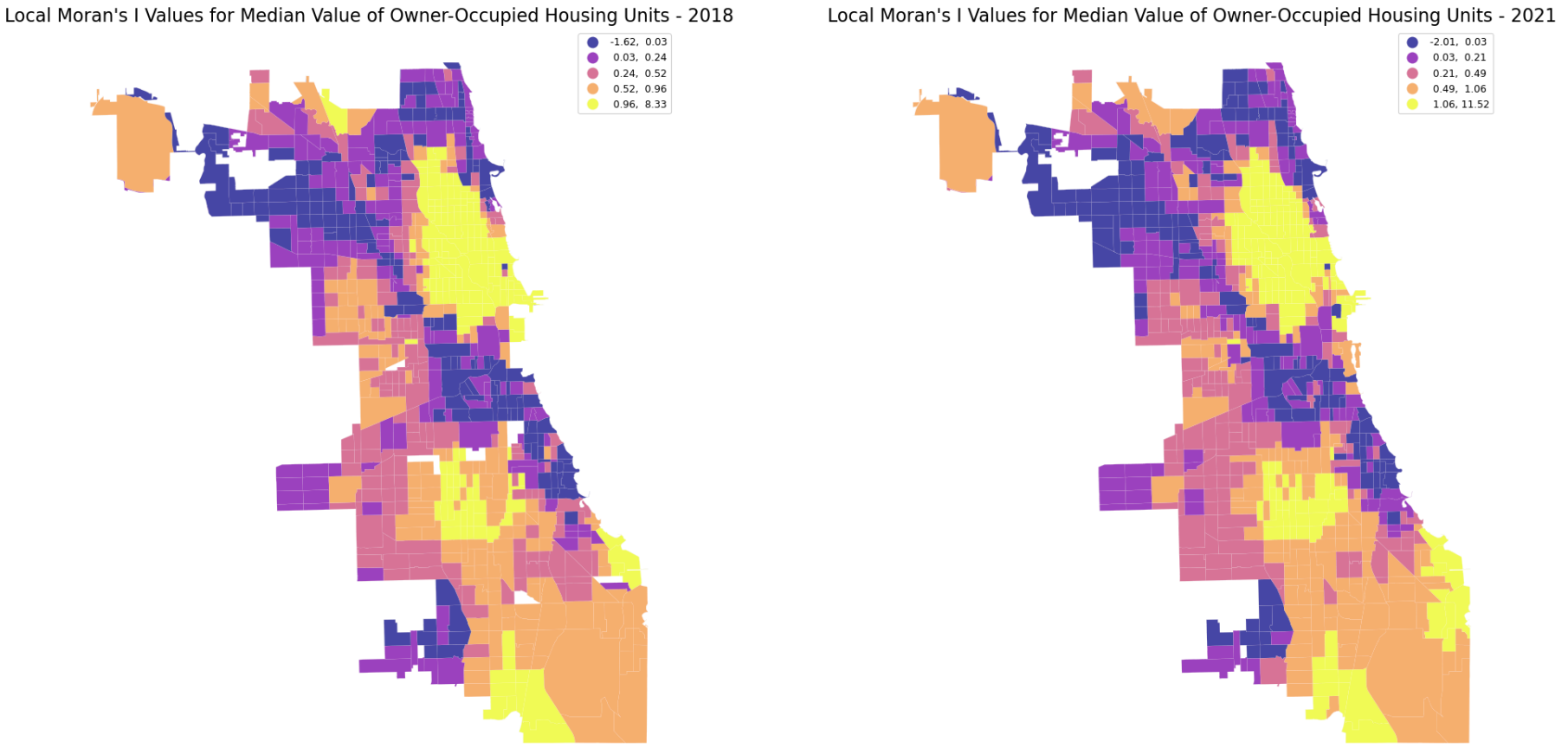




It can be seen that the regions with high housing prices and low housing prices have maintained a relatively stable geographical distribution over the past three years. There has been a price rise in the three years. The housing price ranges in some regions have changed in 2021, but the overall structure remains consistent. High-priced areas are mainly concentrated in the north and along the lake, while the south and southwest remain at lower prices.

In [94]:
```
from libpysal.weights import KNN
from esda.moran import Moran_Local

w_2018 = KNN.from_dataframe(cook_census_geo_2018_filtered_clipped, k=8)
w_2018.transform = "R"
lisa_2018 = Moran_Local(cook_census_geo_2018_filtered_clipped["B25077_001E"], w_2018)

w_2021 = KNN.from_dataframe(cook_census_geo_2021_filtered_clipped, k=8)
w_2021.transform = "R"
lisa_2021 = Moran_Local(cook_census_geo_2021_filtered_clipped["B25077_001E"], w_2021)

fig, axs = plt.subplots(nrows=1, ncols=2, figsize=(24, 12))

cook_census_geo_2018_filtered_clipped.assign(Is=lisa_2018.Is).plot(
    column='Is',
    cmap='plasma',
    scheme='quantiles',
    k=5,
    edgecolor='white',
    linewidth=0.1,
    alpha=0.75,
    legend=True,
    ax=axs[0]
)
axs[0].set_title('Local Moran\'s I Values for Median Value of Owner-Occupied Housing Units - 2018', fontsize=16)
axs[0].set_axis_off()

cook_census_geo_2021_filtered_clipped.assign(Is=lisa_2021.Is).plot(
    column='Is',
    cmap='plasma',
    scheme='quantiles',
    k=5,
    edgecolor='white',
    linewidth=0.1,
    alpha=0.75,
```

```
    legend=True,
    ax=axs[1]
)
axs[1].set_title('Local Moran\'s I Values for Median Value of Owner-Occupied Housing Units - 2021', fontsize=16)
axs[1].set_axis_off()

# 显示整个图形
plt.show()
```



The local Moran's I statistical values of the median values of homeowners occupying housing units in the Chicago area in 2018 and 2021, were used to evaluate the spatial autocorrelation of housing prices. The dark purple area indicates a significant difference in housing prices compared to the surrounding area, while the yellow area indicates that housing prices are similar or clustered with the surrounding area. From 2018 to 2021, the coverage and intensity of positive Moran's I values (yellow areas) have increased, especially in the south, indicating an increase in spatial clustering of housing prices, with regions with similar housing prices becoming more concentrated. This change may reflect changes in market trends or the centralization of real estate development, providing important spatial insights for policy formulation and market analysis.

5.2 Transit Routes and Stations in Chicago

In [20]:
```
# Load 2023 CTA Bus Routes
cta_bus_routes = gpd.read_file(r"data\2023_CTA_Bus_Routes.geojson")
# Load 2023 CTA Bus Stops
cta_bus_stops = gpd.read_file(r"data\2023_CTA_Bus_Stops.geojson")
# Load 2023 CTA Rail Lines
cta_rail_lines = gpd.read_file(r"data\2023_CTA_Rail_Lines.geojson")
# Load 2023 CTA Rail Stations
cta_rail_stations = gpd.read_file(r"data\2023_CTA_Rail_Stations.geojson")
# Load 2023 Metra Rail Stations
metra_rail_stations = gpd.read_file(r"data\2023_Metra_Rail_Stations.geojson")
# Load 2023 Metra Rail Lines
metra_rail_lines = gpd.read_file(r"data\2023_Metra_Rail_Lines.geojson")
# Load 2023 Pace Bus Stops
```

```
pace_bus_stops = gpd.read_file(r"data\2023_Pace_Bus_Stops.geojson")
# Load 2023 Pace Bus Routes
pace_bus_routes = gpd.read_file(r"data\2023_Pace_Bus_Routes.geojson")
# Load Boundaries - Neighborhoods
neighborhood_boundaries = gpd.read_file(r"data\Boundaries - Neighborhoods.geojson")
```

5.2.1 CTA Bus Route and Stops

CTA bus routes and stops are located across the whole city and very wide spread. They also have the highest density among all public transits.

In [21]:
```
cta_bus_routes.head(1)
```

Out[21]:

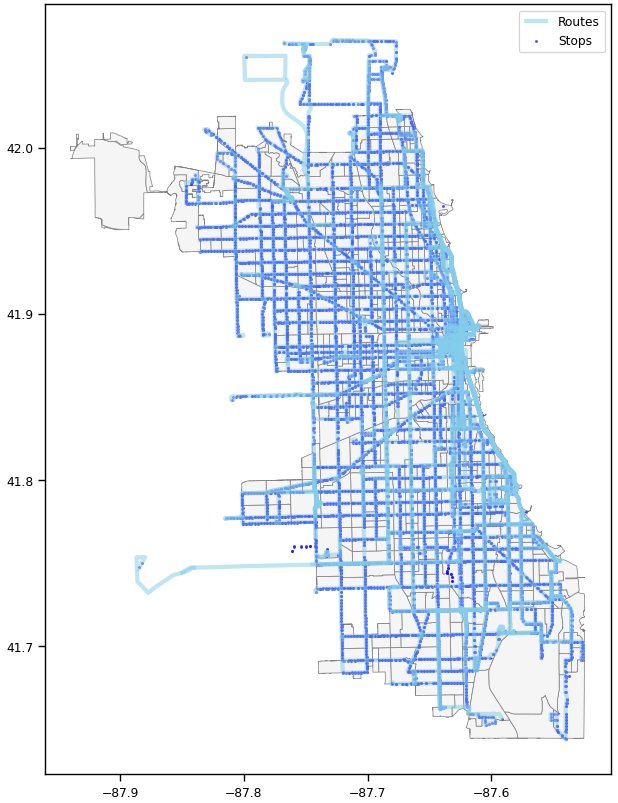
	FID	ROUTE	ROUTE0	NAME	WKDAY	SAT	SUN	SHAPE_LEN	Shape__Length	geometry
0	1	12	012	ROOSEVELT	1	1	1	62586.111408	25595.796952	MULTILINESTRING ((-87.64792 41.86714, -87.6474...

In [22]:
```
cta_bus_stops.head(1)
```

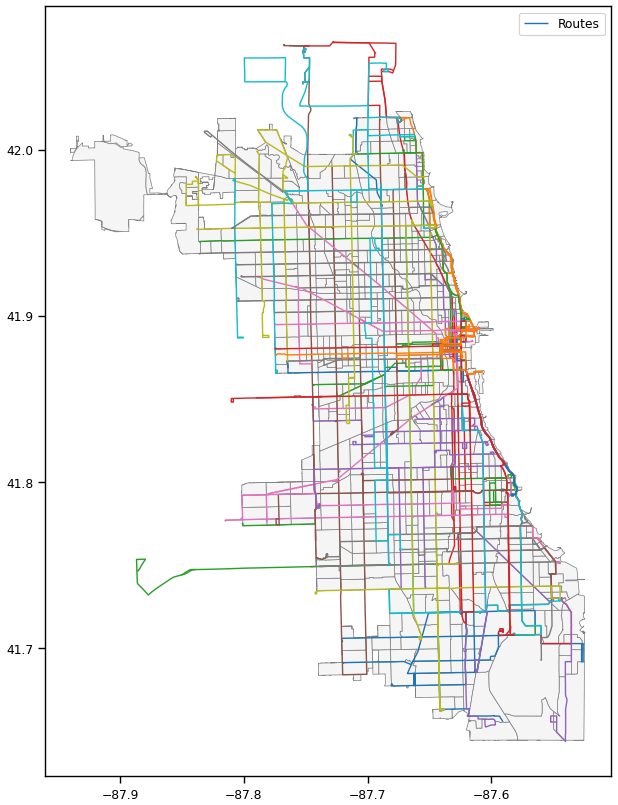
Out[22]:

	FID	SYSTEMSTOP	STREET	CROSS_ST	DIR	POS	ROUTESSTPG	OWLROUTES	CITY	PUBLIC_NAM	geometry
0	1	3847	LARAMIE	QUINCY	SB	NS	57		CHICAGO	Laramie & Quincy	POINT (-87.75503 41.87764)

In [23]:
```
# Create a figure and axis object
fig, ax = plt.subplots(figsize=(10, 10))
# Visualize boundary
cook_census_geo_2021_filtered_clipped.plot(ax=ax, color='#f5f5f5', edgecolor='grey', linewidth=0.5, label='Boundary')
# Visualize routes data (placed behind stops)
cta_bus_routes.plot(ax=ax, color='skyblue', linewidth=3,alpha=0.5,label='Routes')
# Visualize stops data (placed in front of routes)
cta_bus_stops.plot(ax=ax, color='blue', markersize=2, alpha=0.5,label='Stops')
# Add legend
ax.legend()
# Show the plot
plt.show()
```



In [24]:
```
# Create a figure and axis object
fig, ax = plt.subplots(figsize=(10, 10))
# Visualize boundary
cook_census_geo_2021_filtered_clipped.plot(ax=ax, color='#f5f5f5', edgecolor='grey', linewidth=0.5, label='Boundary')
# Visualize routes data (color by 'ROUTE' field)
cta_bus_routes.plot(ax=ax, column='ROUTE', linewidth=1, legend=True, zorder=1, label='Routes')
# Add legend
ax.legend()
# Show the plot
plt.show()
```



5.2.2 CTA Rail Lines and Stations

CTA rail lines and stations are located scarcely in the north west of the city, mainly concentrated in the Loop (City Center).

In [25]:
```
cta_rail_lines.head(1)
```

Out[25]:

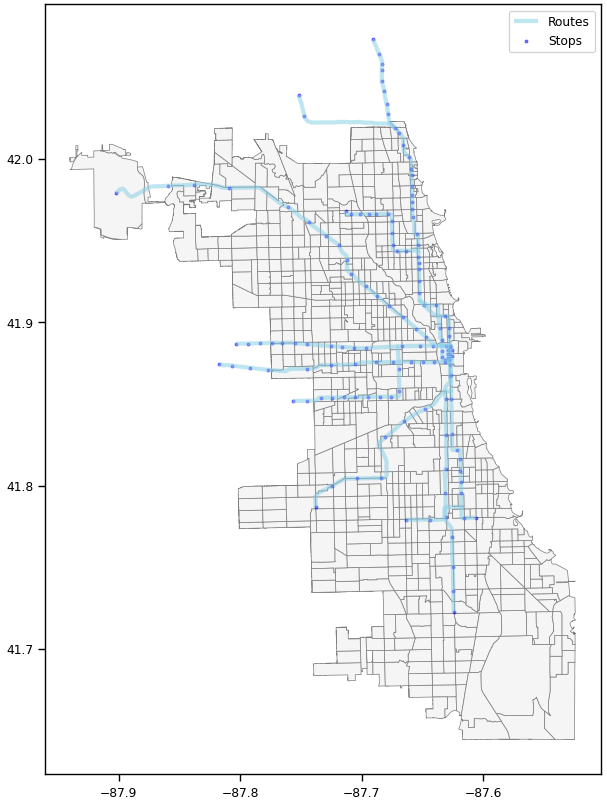
	FID	LINES	DESCRIPTIO	TYPE	LEGEND	SHAPE_LEN	Shape__Length	geometry
0	1	Brown, Orange, Pink, Purple (Express)	Tower 12 to Library	2	ML	647.793225	264.892948	LINESTRING (-87.62821 41.87692, -87.62757 41.8...

In [26]:
```
cta_rail_stations.head(1)
```

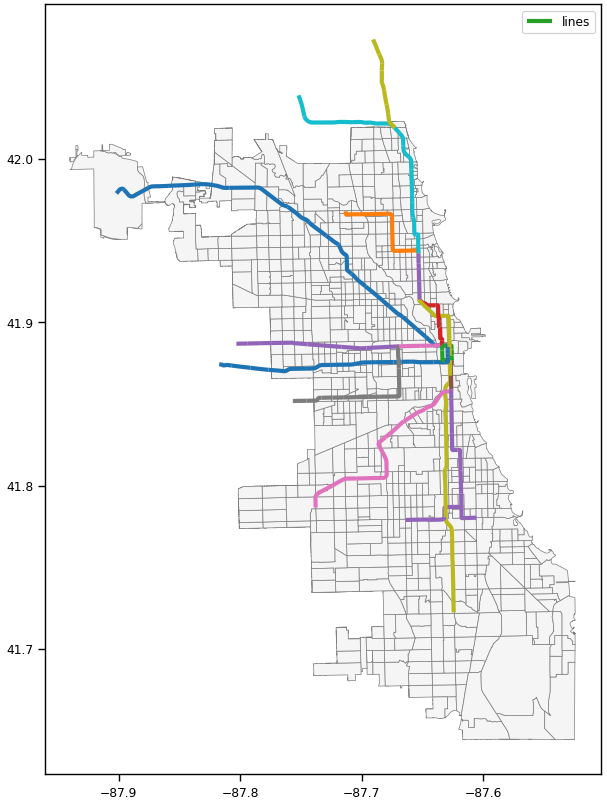
Out[26]:

	FID	STATION_ID	LONGNAME	LINES	ADDRESS	ADA	PKNRD	POINT_X	POINT_Y	geometry
0	1	230	Cumberland	Blue Line	5800 N. Cumberland Avenue	1	1	1.118914e+06	1.937256e+06	POINT (-87.83803 41.98430)

In [27]:
```
# Create a figure and axis object
fig, ax = plt.subplots(figsize=(10, 10))
# Visualize boundary
cook_census_geo_2021_filtered_clipped.plot(ax=ax, color='#f5f5f5', edgecolor='grey', linewidth=0.5, label='Boundary')
# Visualize routes data (placed behind stops)
cta_rail_lines.plot(ax=ax, color='skyblue', linewidth=3,alpha=0.5,label='Routes')
# Visualize stops data (placed in front of routes)
cta_rail_stations.plot(ax=ax, color='blue', markersize=4, alpha=0.5,label='Stops')
# Add legend
ax.legend()
# Show the plot
plt.show()
```



In [28]:
```
# Create a figure and axis object
fig, ax = plt.subplots(figsize=(10, 10))
# Visualize boundary
cook_census_geo_2021_filtered_clipped.plot(ax=ax, color='#f5f5f5', edgecolor='grey', linewidth=0.5, label='Boundary')
# Visualize routes data (color by 'ROUTE' field)
cta_rail_lines.plot(ax=ax, column='LINES', linewidth=3, legend=True, zorder=1, label='lines')
# Add legend
ax.legend()
# Show the plot
plt.show()
```



5.2.3 Pace Bus Route and Stops

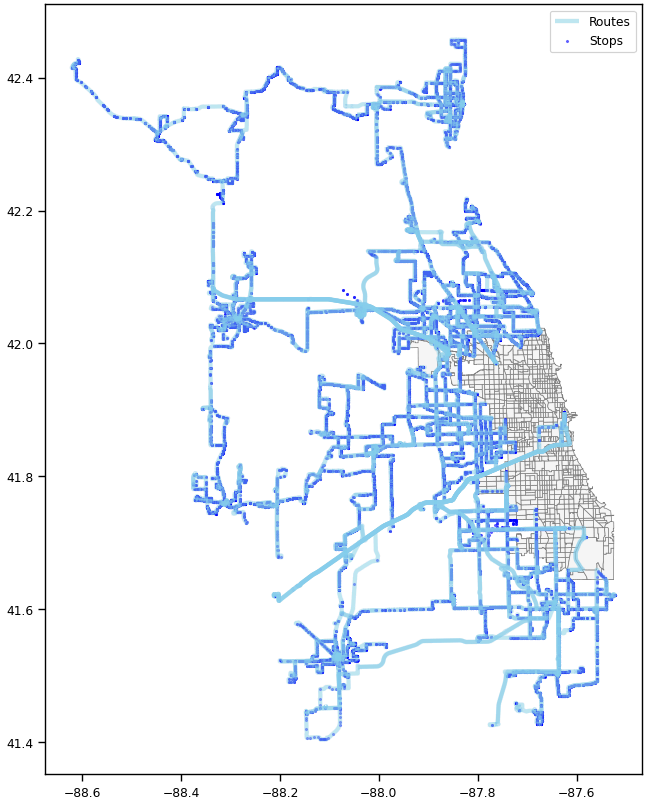
Pace bus route and stops are located in the outskirt of the city, linking the suburbs.

In [29]:
```
pace_bus_routes.head(1)
```

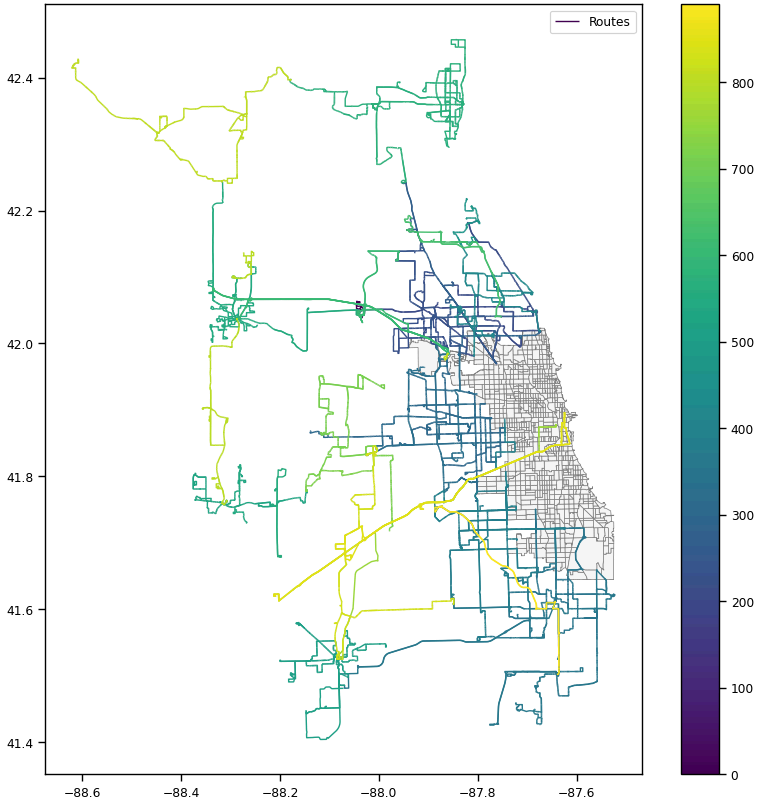
Out[29]:

	FID	route_shor	Shape_Leng	Shape__Length	geometry
0	1	0	0.317273	41451.130361	MULTILINESTRING ((-87.76291 41.97037, -87.7627...

In [30]:
```
# Create a figure and axis object
fig, ax = plt.subplots(figsize=(10, 10))
# Visualize boundary
cook_census_geo_2021_filtered_clipped.plot(ax=ax, color='#f5f5f5', edgecolor='grey', linewidth=0.5, label='Boundary')
# Visualize routes data (placed behind stops)
pace_bus_routes.plot(ax=ax, color='skyblue', linewidth=3,alpha=0.5,label='Routes')
# Visualize stops data (placed in front of routes)
pace_bus_stops.plot(ax=ax, color='blue', markersize=2, alpha=0.5,label='Stops')
# Add legend
ax.legend()
# Show the plot
plt.show()
```



In [31]:
```
# Create a figure and axis object
fig, ax = plt.subplots(figsize=(10, 10))
# Visualize boundary
cook_census_geo_2021_filtered_clipped.plot(ax=ax, color='#f5f5f5', edgecolor='grey', linewidth=0.5, label='Boundary')
# Visualize routes data (color by 'ROUTE' field)
pace_bus_routes.plot(ax=ax, column='route_shor', linewidth=1, legend=True, zorder=1, label='Routes')
# Add legend
ax.legend()
# Show the plot
plt.show()
```



5.2.4 Metra Rail and Lines

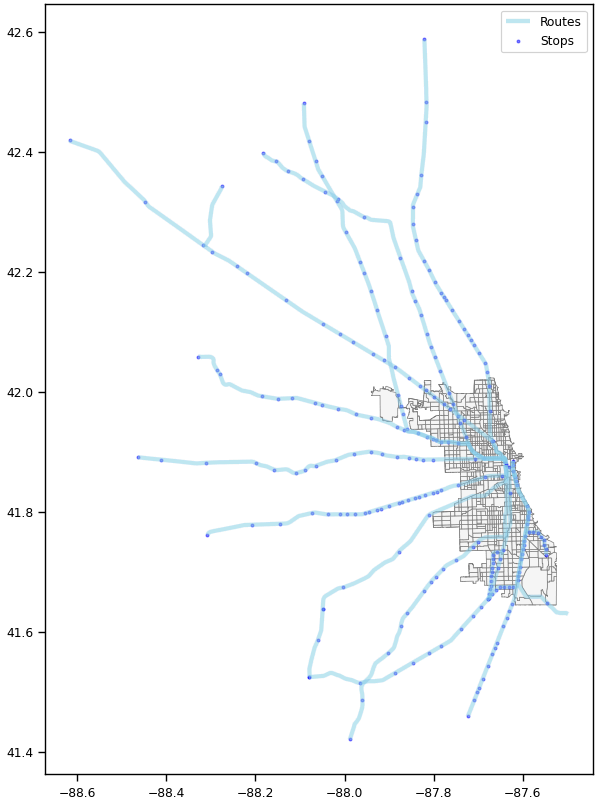
Metra rail and lines are passing through the city from all directions linking the suburbs.

In [32]:
```
metra_rail_lines.head(1)
```

Out[32]:

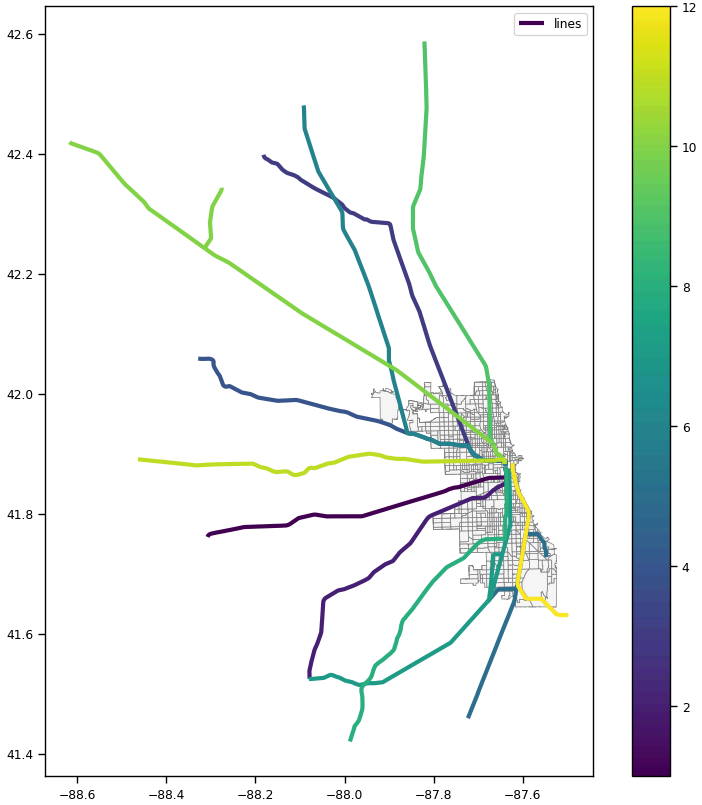
	FID	NAME	ASSET_ID	SHAPE_LEN	Shape__Length	geometry
0	1	BNSF	22200006	195451.738995	79835.207556	LINESTRING (-87.63884 41.87796, -87.63882 41.8...

In [33]:
```
# Create a figure and axis object
fig, ax = plt.subplots(figsize=(10, 10))
# Visualize boundary
cook_census_geo_2021_filtered_clipped.plot(ax=ax, color='#f5f5f5', edgecolor='grey', linewidth=0.5, label='Boundary')
# Visualize routes data (placed behind stops)
metra_rail_lines.plot(ax=ax, color='skyblue', linewidth=3,alpha=0.5,label='Routes')
# Visualize stops data (placed in front of routes)
metra_rail_stations.plot(ax=ax, color='blue', markersize=4, alpha=0.5,label='Stops')
# Add legend
ax.legend()
# Show the plot
plt.show()
```



In [34]:

```
# Create a figure and axis object
fig, ax = plt.subplots(figsize=(10, 10))
# Visualize boundary
cook_census_geo_2021_filtered_clipped.plot(ax=ax, color='#f5f5f5', edgecolor='grey', linewidth=0.5, label='Boundary')
# Visualize routes data (color by 'ROUTE' field)
metra_rail_lines.plot(ax=ax, column='FID', linewidth=3, legend=True, zorder=1, label='lines')
# Add legend
ax.legend()
# Show the plot
plt.show()
```



5.3 Data Merging of Ridership Data

Because the CTA and Metra Rail datasets are the only ones that provide station-by-station ridership figures, however, Metra Rail contributes more contribute to suburbs rather than the city of Chicago, therefore, only CTA data is chosen to be analyzed in this project. Additionally, the average weekday, Saturday, and Sunday's ridership across all months of CTA Bus is calculated. It is important to note that the CTA dataset is the only one that includes data from both 2018 and 2021.

5.3.1 CTA Ridership Data

In [35]:
```
file_path = r"data\CTA_Average_Rail_Station_Ridership_1999_2023.csv"
cta_ridership = pd.read_csv(file_path)
cta_ridership_2018 = cta_ridership[(cta_ridership['YEAR'] == 2018) & (cta_ridership['DAY_TYPE'] == 'Weekday')]
cta_ridership_2018 = cta_ridership_2018.groupby('RIDERSHIP_ID').agg({'DAILY_AVG_RIDES': 'mean'})
cta_ridership_2018.reset_index(inplace=True)
cta_ridership_2018.rename(columns={'RIDERSHIP_ID': 'STATION_ID'}, inplace=True)
cta_ridership_2018.head()
```

Out[35]:

	STATION_ID	DAILY_AVG_RIDES
0	10	1934.250000
1	20	3746.916667
2	30	1396.000000
3	40	7961.416667
4	50	3639.750000

In [36]:
```
file_path = r"data\CTA_Average_Rail_Station_Ridership_1999_2023.csv"
cta_ridership = pd.read_csv(file_path)
cta_ridership_2021 = cta_ridership[(cta_ridership['YEAR'] == 2021) & (cta_ridership['DAY_TYPE'] == 'Weekday')]
cta_ridership_2021 = cta_ridership_2021.groupby('RIDERSHIP_ID').agg({'DAILY_AVG_RIDES': 'mean'})
cta_ridership_2021.reset_index(inplace=True)
```

```
cta_ridership_2021.rename(columns={'RIDERSHIP_ID': 'STATION_ID'}, inplace=True)
cta_ridership_2021.head()
```

Out[36]:

	STATION_ID	DAILY_AVG_RIDES
0	10	551.250000
1	20	1404.833333
2	30	657.916667
3	40	1703.333333
4	50	1163.083333

5.3.2 Metra Ridership Data

In [37]:
```
file_path = r"data\Metra_Ridership_by_Station_Boarding_Alighting Survey_1979_2018_0.csv"
metra_ridership = pd.read_csv(file_path)
metra_ridership_2018 = metra_ridership[(metra_ridership['YEAR'] == 2018)]
metra_ridership_2018 = metra_ridership_2018.groupby('STATION').agg({'BOARDS': 'mean','ALIGHTS': 'mean'})
metra_ridership_2018.head()
```

Out[37]:

STATION	BOARDS	ALIGHTS
103rd Street (Beverly Hills)	734.0	731.0
103rd Street (Rosemoor)	36.0	34.0
107th Street	27.0	23.0
107th Street (Beverly Hills)	395.0	368.0
111th Street	31.0	49.0

Join ridership data to the subway station data

In [38]:
```
cta_rail_stations.head(1)
```

Out[38]:

	FID	STATION_ID	LONGNAME	LINES	ADDRESS	ADA	PKNRD	POINT_X	POINT_Y	geometry
0	1	230	Cumberland	Blue Line	5800 N. Cumberland Avenue	1	1	1.118914e+06	1.937256e+06	POINT (-87.83803 41.98430)

In [39]:
```
cta_rail_stations.rename(columns={'LONGNAME': 'NAME'}, inplace=True)
```

In [40]:
```
cta_rail_stations.head(1)
```

Out[40]:

	FID	STATION_ID	NAME	LINES	ADDRESS	ADA	PKNRD	POINT_X	POINT_Y	geometry
0	1	230	Cumberland	Blue Line	5800 N. Cumberland Avenue	1	1	1.118914e+06	1.937256e+06	POINT (-87.83803 41.98430)

In [41]:
```
cta_rail_ridership = pd.merge(cta_rail_stations, cta_ridership_2018[['STATION_ID', 'DAILY_AVG_RIDES']], left_on='STATION_ID', right_on='STATION_ID', how='left')
cta_rail_ridership.rename(columns={'DAILY_AVG_RIDES': 'RIDE2018'}, inplace=True)
cta_rail_ridership = pd.merge(cta_rail_ridership, cta_ridership_2021[['STATION_ID', 'DAILY_AVG_RIDES']], left_on='STATION_ID', right_on='STATION_ID', how='left')
cta_rail_ridership.rename(columns={'DAILY_AVG_RIDES': 'RIDE2021'}, inplace=True)
cta_rail_ridership.head()
```

Out[41]:

	FID	STATION_ID	NAME	LINES	ADDRESS	ADA	PKNRD	POINT_X	POINT_Y	geometry	RIDE2018	RIDE2021
0	1	230	Cumberland	Blue Line	5800 N. Cumberland Avenue	1	1	1.118914e+06	1.937256e+06	POINT (-87.83803 41.98430)	4498.583333	1177.333333
1	2	1350	Oak Park-Lake	Green Line (Lake)	100 S. Oak Park Avenue	0	0	1.131166e+06	1.901870e+06	POINT (-87.79379 41.88700)	1578.583333	426.166667
2	3	1260	Austin-Lake	Green Line (Lake)	351 N. Austin Blvd	0	0	1.136515e+06	1.902016e+06	POINT (-87.77414 41.88730)	1822.916667	701.583333
3	4	170	Ashland-Lake	Green (Lake), Pink	1601 W. Lake Street	1	0	1.165707e+06	1.901502e+06	POINT (-87.66696 41.88532)	2431.750000	992.583333
4	5	1160	Clinton-Lake	Green (Lake), Pink	540 W. Lake Street	1	0	1.172562e+06	1.901699e+06	POINT (-87.64178 41.88571)	4406.916667	1247.250000

In [42]:
```
cta_rail_ridership['railscale'] = cta_rail_ridership['RIDE2021'] / cta_rail_ridership['RIDE2018']
cta_rail_ridership.head()
```

Out[42]:

	FID	STATION_ID	NAME	LINES	ADDRESS	ADA	PKNRD	POINT_X	POINT_Y	geometry	RIDE2018	RIDE2021	railscale
0	1	230	Cumberland	Blue Line	5800 N. Cumberland Avenue	1	1	1.118914e+06	1.937256e+06	POINT (-87.83803 41.98430)	4498.583333	1177.333333	0.261712
1	2	1350	Oak Park-Lake	Green Line (Lake)	100 S. Oak Park Avenue	0	0	1.131166e+06	1.901870e+06	POINT (-87.79379 41.88700)	1578.583333	426.166667	0.269968
2	3	1260	Austin-Lake	Green Line (Lake)	351 N. Austin Blvd	0	0	1.136515e+06	1.902016e+06	POINT (-87.77414 41.88730)	1822.916667	701.583333	0.384869
3	4	170	Ashland-Lake	Green (Lake), Pink	1601 W. Lake Street	1	0	1.165707e+06	1.901502e+06	POINT (-87.66696 41.88532)	2431.750000	992.583333	0.408177
4	5	1160	Clinton-Lake	Green (Lake), Pink	540 W. Lake Street	1	0	1.172562e+06	1.901699e+06	POINT (-87.64178 41.88571)	4406.916667	1247.250000	0.283021

Join ridership data to the bus line data

In [43]:
```
cta_bus_routes.head(1)
```

Out[43]:

	FID	ROUTE	ROUTE0	NAME	WKDAY	SAT	SUN	SHAPE_LEN	Shape__Length	geometry
0	1	12	012	ROOSEVELT	1	1	1	62586.111408	25595.796952	MULTILINESTRING ((-87.64792 41.86714, -87.6474...

In [44]:
```
#cta_bus_routes['ROUTE'].unique()
```

In [45]:
```
#cta_bus_routes['ROUTE0'].unique()
```

In [46]:
```
file_path = r"data\CTA_Average_Bus_Ridership_1999_2023.csv"
bus_ridership = pd.read_csv(file_path)
bus_ridership['ROUTE'].unique()
```

Out[46]:
```
array(['0', '1', '100', '1001', '103', '106', '108', '11', '111', '111A',
       '112', '115', '119', '12', '120', '121', '124', '125', '126',
       '128', '134', '135', '136', '143', '146', '147', '148', '15',
       '151', '152', '155', '156', '157', '165', '169', '171', '172',
       '18', '19', '192', '2', '20', '201', '206', '21', '22', '24', '26',
       '28', '29', '3', '30', '31', '34', '35', '36', '37', '39', '4',
       '43', '44', '47', '48', '49', '49B', '5', '50', '51', '52', '52A',
       '53', '53A', '54', '54A', '54B', '55', '55A', '55N', '56', '57',
       '59', '6', '60', '62', '62H', '63', '63W', '65', '66', '67', '68',
       '7', '70', '71', '72', '73', '74', '75', '76', '77', '78', '79',
       '8', '80', '81', '81W', '82', '84', '85', '85A', '86', '87', '88',
       '8A', '9', '90', '91', '92', '93', '94', '95', '96', '97', 'J14',
       'X4', 'X49', 'X9', '10', '130', 'X98', '642', '132', '205', '95E',
       '95W', '170', '154', '33', 'R22', 'R39', 'R55', 'R63', 'R69',
       'R79', 'R87', 'R95', '122', '123', '129', '144', '145', '17',
       '49A', '56A', '64', '69', '90N', 'N201', 'X28', '14', '168',
       '53AL', 'X20', 'X3', 'X54', 'X55', 'X80', '174', '200', '38',
       '173', '127', '1002', '25', '40', 'X99', 'X21', '27', '202', '203',
       '204', 'N5', '1003'], dtype=object)
```

In [47]:
```
bus_ridership.head()
```

Out[47]:

	ROUTE	YEAR	MONTH	DAY_TYPE	AVG_RIDES
0	0	2023	10	Weekday	2626
1	0	2023	10	Saturday	4694
2	0	2023	10	Sunday - Holiday	2634
3	1	2023	10	Weekday	1126
4	100	2023	10	Weekday	470

Calculate CTA bus ridership by weekday, saturday, and sunday

In [48]:
```
# Filter the data for the years 2018 and 2021
ridership_2018 = bus_ridership[bus_ridership['YEAR'] == 2018]
ridership_2021 = bus_ridership[bus_ridership['YEAR'] == 2021]

# Calculate the average monthly ridership by day type for each route
avg_ridership_2018 = ridership_2018.groupby(['ROUTE', 'DAY_TYPE'])['AVG_RIDES'].mean().unstack()
avg_ridership_2021 = ridership_2021.groupby(['ROUTE', 'DAY_TYPE'])['AVG_RIDES'].mean().unstack()

# Rename the columns as per your requirement
avg_ridership_2018.columns = ['saturday2018', 'sunday2018', 'weekday2018']
avg_ridership_2021.columns = ['saturday2021', 'sunday2021', 'weekday2021']
```

In [49]:
```
avg_ridership_2018.head()
```

Out[49]:

ROUTE	saturday2018	sunday2018	weekday2018
0	2710.833333	1586.50	3680.833333
1	NaN	NaN	1709.833333
10	455.600000	473.60	521.000000
100	NaN	NaN	569.666667
1001	139.083333	121.75	85.916667

In [50]:
```
avg_ridership_2021.head()
```

Out[50]:

ROUTE	saturday2021	sunday2021	weekday2021
0	2079.666667	1540.250000	3063.500000
1	NaN	NaN	376.833333
10	408.000000	331.400000	362.250000
100	NaN	NaN	243.416667
1001	463.750000	495.583333	53.416667

Merge 2 years ridership data together

In [51]:
```
cta_bus_routes = pd.merge(cta_bus_routes, avg_ridership_2018, left_on='ROUTE', right_index=True, how='left')
cta_bus_routes = pd.merge(cta_bus_routes, avg_ridership_2021, left_on='ROUTE', right_index=True, how='left')
```

In [52]:
```
cta_bus_routes.head()
```

Out[52]:

	FID	ROUTE	ROUTE0	NAME	WKDAY	SAT	SUN	SHAPE_LEN	Shape__Length	geometry	saturday2018	sunday2018	weekday2018	saturday2021	sunday2021	weekday2021
0	1	12	012	ROOSEVELT	1	1	1	62586.111408	25595.796952	MULTILINESTRING ((-87.64792 41.86714, -87.6474...	7589.083333	5757.416667	12381.416667	3761.750000	2735.916667	5725.083333
1	2	121	121	UNION/STREETERVILLE EXPRESS	1	0	0	24090.823988	9866.558032	MULTILINESTRING ((-87.62451 41.88834, -87.6249...	NaN	NaN	1111.666667	NaN	NaN	130.250000
2	3	1	001	BRONZEVILLE/UNION STATION	1	0	0	34690.953676	14214.349988	MULTILINESTRING ((-87.62327 41.83104, -87.6232...	NaN	NaN	1709.833333	NaN	NaN	376.833333
3	4	108	108	HALSTED/95TH	1	0	0	71888.990825	29369.622395	MULTILINESTRING ((-87.59054 41.65578, -87.5906...	NaN	NaN	1138.500000	NaN	NaN	380.583333
4	5	11	011	LINCOLN	1	1	1	24694.573889	10138.959924	MULTILINESTRING ((-87.68884 41.96677, -87.6888...	799.583333	528.000000	1452.333333	491.416667	337.166667	767.000000

Calculate the weekend bus ridership

In [53]:
```
# Replace NaN values with 0
cta_bus_routes[['saturday2018', 'sunday2018', 'weekday2018', 'saturday2021', 'sunday2021', 'weekday2021']] = cta_bus_routes[['saturday2018', 'sunday2018', 'weekday2018', 'saturday2021', 'sunday2021', 'weekday2021']].fillna(0)
cta_bus_routes['weekend2018'] = cta_bus_routes['saturday2018'] + cta_bus_routes['sunday2018']
cta_bus_routes['weekend2021'] = cta_bus_routes['saturday2021'] + cta_bus_routes['sunday2021']
# Calculate total ridership for each year
cta_bus_routes['total2018'] = cta_bus_routes['saturday2018'] + cta_bus_routes['sunday2018'] + cta_bus_routes['weekday2018']
cta_bus_routes['total2021'] = cta_bus_routes['saturday2021'] + cta_bus_routes['sunday2021'] + cta_bus_routes['weekday2021']

# Calculate the scale
cta_bus_routes['busscale'] = cta_bus_routes['total2021'] / cta_bus_routes['total2018']
cta_bus_routes.head()
```

Out[53]:

	FID	ROUTE	ROUTE0	NAME	WKDAY	SAT	SUN	SHAPE_LEN	Shape__Length	geometry	...	sunday2018	weekday2018	saturday2021	sunday2021	weekday2021	weekend2018	weekend2021	total2018	total2021	busscale
0	1	12	012	ROOSEVELT	1	1	1	62586.111408	25595.796952	MULTILINESTRING ((-87.64792 41.86714, -87.6474...	...	5757.416667	12381.416667	3761.750000	2735.916667	5725.083333	13346.500000	6497.666667	25727.916667	12222.750000	0.475077
1	2	121	121	UNION/STREETERVILLE EXPRESS	1	0	0	24090.823988	9866.558032	MULTILINESTRING ((-87.62451 41.88834, -87.6249...	...	0.000000	1111.666667	0.000000	0.000000	130.250000	0.000000	0.000000	1111.666667	130.250000	0.117166
2	3	1	001	BRONZEVILLE/UNION STATION	1	0	0	34690.953676	14214.349988	MULTILINESTRING ((-87.62327 41.83104, -87.6232...	...	0.000000	1709.833333	0.000000	0.000000	376.833333	0.000000	0.000000	1709.833333	376.833333	0.220392
3	4	108	108	HALSTED/95TH	1	0	0	71888.990825	29369.622395	MULTILINESTRING ((-87.59054 41.65578, -87.5906...	...	0.000000	1138.500000	0.000000	0.000000	380.583333	0.000000	0.000000	1138.500000	380.583333	0.334285
4	5	11	011	LINCOLN	1	1	1	24694.573889	10138.959924	MULTILINESTRING ((-87.68884 41.96677, -87.6888...	...	528.000000	1452.333333	491.416667	337.166667	767.000000	1327.583333	828.583333	2779.916667	1595.583333	0.573968

5 rows × 21 columns

6 Models

6.1 Variables

General Variables

`B02001_001E` : Total population by race (abandoned)

`nonwhite_perc`

`foreign_perc`

`vacant_unit_perc`

`owner_occupied_perc` (abandoned)

`renter_occupied_perc` (abandoned)

`B25077_001E` : Median value (dollars) of owner-occupied housing units.

`B25085_001E` : Price asked for vacant-for-sale-only and sold, not occupied housing units. (abandoned)

`RIDE2018` : 2018 average daily Subway ridership by station

`RIDE2021` : 2021 average daily Subway ridership by station

`railscale/busscale` : Total ridership in 2021 devided by 2018's ridership (abandoned)

`saturday2018` : 2018 average monthly saturday bus ridership by line (abandoned)

`sunday2018` : 2018 average monthly sunday/holiday bus ridership by line (abandoned)

`weekend2018` : 2018 average monthly weekend bus ridership by line (abandoned)

`weekday2018` : 2018 average monthly weekday bus ridership by line (abandoned)

`saturday2021` : 2021 average monthly saturday bus ridership by line (abandoned)

`sunday2021` : 2021 average monthly sunday/holiday bus ridership by line (abandoned)

`weekday2021` : 2021 average monthly weekday bus ridership by line (abandoned)

`weekend2021` : 2021 average monthly weekend bus ridership by line (abandoned)

`total2018` : 2018 average monthly bus ridership by line (abandoned)

`total2021` : 2021 average monthly bus ridership by line (abandoned)

New Variables

`logPop` : Log `B02001_001E` (abandoned)

`logMidHomeValue` : log `B25077_001E`

`logSaleHouseValue` : log `B25085_001E` (abandoned)

Distance Variables

Distance from Tracts' to Cloest CTA bus stop

Distance from Tracts' to Cloest CTA rail stop

Distance from Tracts' to UChicago [-87.60093, 41.78958] pri (abandoned)

Distance from Tracts' to UIC [-87.65100, 41.87366] pub (abandoned)

Distance from Tracts' to Northwestern [-87.67708, 42.05437] pri (abandoned)

Distance from Tracts' to NEIU [-87.72031, 41.97870] pub (abandoned)

Distance from Tracts' to Illinois Tech [-87.62721, 41.83139] pri (abandoned)

Distance from Tracts' to DePaul (Loop) [-87.62723, 41.87775] pub (abandoned)

Distance from Tracts' to DePaul (Lincoln Park) [-87.65878, 41.92500] pub (abandoned)

Distance from Tracts' to Loyola (Lake Shore Campus) [-87.65779, 41.99858] pri (abandoned)

Distance from Tracts' to Loyola (Health Sciences Campus) [-87.83338, 41.86515] pri (abandoned)

Distance from Tracts' to Loyola (Water Tower Campus) [-87.62517, 41.89733] pri (abandoned)

Distance from Tracts' to Chicago State [-87.61350, 41.73000] pub (abandoned)

Distance from Tracts' to CBD (abandoned)

Relative variables

Dummy (is_private) : if the closest university to this tract is a private university [1,0] (abandoned)

Dummy Weekday-only bus line, use SAT [1,0] (abandoned)

Dummy Subway with ADA (elevator) [0,1] (abandoned)

Dummy Subway with PKNRD (Parking Pool) [0,1] (abandoned)

Dummy (is_CBD) : if the tract is CBD [1,0]

6.2 Data Processing

Calculate log variables

In [54]:
```
# Replace 0 with 1 in the specified columns of acs_df_2018 and acs_df_2021
acs_df_2018['B02001_001E'] = acs_df_2018['B02001_001E'].replace(0, 1)
acs_df_2021['B02001_001E'] = acs_df_2021['B02001_001E'].replace(0, 1)
acs_df_2018['B25077_001E'] = acs_df_2018['B25077_001E'].replace(0, 1)
acs_df_2021['B25077_001E'] = acs_df_2021['B25077_001E'].replace(0, 1)
acs_df_2018['B25085_001E'] = acs_df_2018['B25085_001E'].replace(0, 1)
acs_df_2021['B25085_001E'] = acs_df_2021['B25085_001E'].replace(0, 1)

# Calculate the logarithm of total population
acs_df_2018['logPop'] = np.log(np.nan_to_num(acs_df_2018['B02001_001E']))
acs_df_2021['logPop'] = np.log(np.nan_to_num(acs_df_2021['B02001_001E']))

# Calculate log of housing values
acs_df_2018['logMidHomeValue'] = np.log(np.nan_to_num(acs_df_2018['B25077_001E']))
acs_df_2021['logMidHomeValue'] = np.log(np.nan_to_num(acs_df_2021['B25077_001E']))
acs_df_2018['logSaleHouseValue'] = np.log(np.nan_to_num(acs_df_2018['B25085_001E']))
acs_df_2021['logSaleHouseValue'] = np.log(np.nan_to_num(acs_df_2021['B25085_001E']))
```

Verify

In [55]:
```
acs_df_2018.head(1)
```

Out[55]:

	NAME	B25077_001E	B25085_001E	B25001_001E	B25002_003E	B25003_002E	B25003_003E	B06011_001E	B05007_001E	B02001_001E	...	tract	GEOID	nonwhite_perc	foreign_perc	vacant_unit_perc	owner_occupied_perc	renter_occupied_perc	logPop	logMidH
0	Census Tract 4302, Cook County, Illinois	439600.0	1.0	2556.0	177.0	341.0	2038.0	18893.0	281.0	4961.0	...	430200	17031430200	0.91534	0.056642	0.069249	0.133412	0.79734	8.509363	

1 rows × 23 columns

In [56]:
```
acs_df_2021.head(1)
```

Out[56]:

	NAME	B25077_001E	B25085_001E	B25001_001E	B25002_003E	B25003_002E	B25003_003E	B06011_001E	B05007_001E	B02001_001E	...	tract	GEOID	nonwhite_perc	foreign_perc	vacant_unit_perc	owner_occupied_perc	renter_occupied_perc	logPop	logMidH
0	Census Tract 101, Cook County, Illinois	255000.0	63.0	2935.0	524.0	808.0	1603.0	33768.0	951.0	4534.0	...	010100	17031010100	0.557786	0.209749	0.178535	0.275298	0.546167	8.41936	

1 rows × 23 columns

In [57]:
```
print(acs_df_2018.columns)
```

```
Index(['NAME', 'B25077_001E', 'B25085_001E', 'B25001_001E', 'B25002_003E',
       'B25003_002E', 'B25003_003E', 'B06011_001E', 'B05007_001E',
       'B02001_001E', 'B02001_002E', 'state', 'county', 'tract', 'GEOID',
       'nonwhite_perc', 'foreign_perc', 'vacant_unit_perc',
       'owner_occupied_perc', 'renter_occupied_perc', 'logPop',
       'logMidHomeValue', 'logSaleHouseValue'],
      dtype='object')
```

Merge the demographic data to geodatabase

In [58]:
```
acs_df_2018 = cook_tract.merge(acs_df_2018, left_on = 'GEOID', right_on = 'GEOID')
acs_df_2018.head()
```

Out[58]:

	STATEFP	COUNTYFP	TRACTCE	GEOID	NAME_x	NAMELSAD	MTFCC	FUNCSTAT	ALAND	AWATER	...	county	tract	nonwhite_perc	foreign_perc	vacant_unit_perc	owner_occupied_perc	renter_occupied_perc	logPop	logMidHomeValue	logSaleHouseValue
0	17	031	221000	17031221000	2210	Census Tract 2210	G5020	S	220053	0	...	031	221000	0.243331	0.169430	0.089847	0.297394	0.612758	7.928046	12.570716	2.302585
1	17	031	221100	17031221100	2211	Census Tract 2211	G5020	S	440042	0	...	031	221100	0.251008	0.288650	0.072526	0.286263	0.641210	8.557759	12.746069	3.044522
2	17	031	242800	17031242800	2428	Census Tract 2428	G5020	S	439630	0	...	031	242800	0.052529	0.127756	0.079812	0.593114	0.327074	7.340836	13.010090	0.000000
3	17	031	242900	17031242900	2429	Census Tract 2429	G5020	S	323819	0	...	031	242900	0.239951	0.136054	0.107987	0.388076	0.503937	7.388328	12.937479	0.000000
4	17	031	243000	17031243000	2430	Census Tract 2430	G5020	S	324547	0	...	031	243000	0.249780	0.174582	0.040075	0.363467	0.596459	7.729296	12.892195	2.397895

5 rows × 35 columns

In [59]:
```
print(acs_df_2018.columns)
```

```
Index(['STATEFP', 'COUNTYFP', 'TRACTCE', 'GEOID', 'NAME_x', 'NAMELSAD',
       'MTFCC', 'FUNCSTAT', 'ALAND', 'AWATER', 'INTPTLAT', 'INTPTLON',
       'geometry', 'NAME_y', 'B25077_001E', 'B25085_001E', 'B25001_001E',
       'B25002_003E', 'B25003_002E', 'B25003_003E', 'B06011_001E',
       'B05007_001E', 'B02001_001E', 'B02001_002E', 'state', 'county', 'tract',
       'nonwhite_perc', 'foreign_perc', 'vacant_unit_perc',
       'owner_occupied_perc', 'renter_occupied_perc', 'logPop',
       'logMidHomeValue', 'logSaleHouseValue'],
      dtype='object')
```

In [60]:
```
acs_df_2021 = cook_tract.merge(acs_df_2021, left_on = 'GEOID', right_on = 'GEOID')
acs_df_2021.head()
```

Out[60]:

	STATEFP	COUNTYFP	TRACTCE	GEOID	NAME_x	NAMELSAD	MTFCC	FUNCSTAT	ALAND	AWATER	...	county	tract	nonwhite_perc	foreign_perc	vacant_unit_perc	owner_occupied_perc	renter_occupied_perc	logPop	logMidHomeValue	logSaleHouseValue
0	17	031	221000	17031221000	2210	Census Tract 2210	G5020	S	220053	0	...	031	221000	0.338911	0.205447	0.085985	0.294927	0.619089	7.851661	12.760819	0.000000
1	17	031	221100	17031221100	2211	Census Tract 2211	G5020	S	440042	0	...	031	221100	0.351490	0.190084	0.109783	0.276671	0.613546	8.411388	12.803259	2.995732
2	17	031	242800	17031242800	2428	Census Tract 2428	G5020	S	439630	0	...	031	242800	0.118514	0.132607	0.065156	0.543909	0.390935	7.353082	13.169438	0.000000
3	17	031	242900	17031242900	2429	Census Tract 2429	G5020	S	323819	0	...	031	242900	0.334938	0.249866	0.135776	0.463362	0.400862	7.533159	12.938441	0.000000
4	17	031	243000	17031243000	2430	Census Tract 2430	G5020	S	324547	0	...	031	243000	0.372104	0.118582	0.068221	0.488774	0.443005	7.696667	13.008747	0.000000

5 rows × 35 columns

Calculate the distance for each tract to the universities

In [61]:
```
acs_df_2018.crs
```

Out[61]:
```
<Geographic 2D CRS: EPSG:4269>
Name: NAD83
Axis Info [ellipsoidal]:
- Lat[north]: Geodetic latitude (degree)
- Lon[east]: Geodetic longitude (degree)
Area of Use:
- name: North America - onshore and offshore: Canada - Alberta; British Columbia; Manitoba; New Brunswick; Newfoundland and Labrador; Northwest Territories; Nova Scotia; Nunavut; Ontario; Prince Edward Island; Quebec; Saskatchewan; Yukon. Puerto Ric
o. United States (USA) - Alabama; Alaska; Arizona; Arkansas; California; Colorado; Connecticut; Delaware; Florida; Georgia; Hawaii; Idaho; Illinois; Indiana; Iowa; Kansas; Kentucky; Louisiana; Maine; Maryland; Massachusetts; Michigan; Minnesota; Miss
issippi; Missouri; Montana; Nebraska; Nevada; New Hampshire; New Jersey; New Mexico; New York; North Carolina; North Dakota; Ohio; Oklahoma; Oregon; Pennsylvania; Rhode Island; South Carolina; South Dakota; Tennessee; Texas; Utah; Vermont; Virginia;
Washington; West Virginia; Wisconsin; Wyoming. US Virgin Islands. British Virgin Islands.
- bounds: (167.65, 14.92, -40.73, 86.45)
Datum: North American Datum 1983
- Ellipsoid: GRS 1980
- Prime Meridian: Greenwich
```

In [62]:
```
places = {
    'UChicago': [-87.60093, 41.78958],
    'UIC': [-87.65100, 41.87366],
    'Northwestern': [-87.67708, 42.05437],
    'NEIU': [-87.72031, 41.97870],
    'Illinois Tech': [-87.62721, 41.83139],
    'DePaul_Loop': [-87.62723, 41.87775],
    'DePaul_Lincoln_Park': [-87.65878, 41.92500],
    'Loyola_Lake_Shore': [-87.65779, 41.99858],
    'Loyola_Health_Sciences': [-87.83338, 41.86515],
    'Loyola_Water_Tower': [-87.62517, 41.89733],
    'Chicago_State': [-87.61350, 41.73000]
}

places_df = gpd.GeoDataFrame(
    {'place': list(places.keys()),
     'geometry': [Point(lon, lat) for lon, lat in places.values()]},
    crs="EPSG:4326"  # 使用WGS 84坐标系
)

acs_df_2018 = acs_df_2018.to_crs(epsg=3857)
places_df = places_df.to_crs(epsg=3857)

for place in places:
    acs_df_2018[f'distance_to_{place}'] = acs_df_2018.geometry.apply(
        lambda x: x.distance(places_df.loc[places_df['place'] == place, 'geometry'].squeeze())
    )

acs_df_2021 = acs_df_2021.to_crs(epsg=3857)
places_df = places_df.to_crs(epsg=3857)

for place in places:
    acs_df_2021[f'distance_to_{place}'] = acs_df_2021.geometry.apply(
        lambda x: x.distance(places_df.loc[places_df['place'] == place, 'geometry'].squeeze())
    )
```

Validate the distance to universities' result

In [63]:
```
acs_df_2018[['GEOID', 'INTPTLAT', 'INTPTLON', 'logPop',
       'logMidHomeValue', 'logSaleHouseValue', 'distance_to_UChicago',
       'distance_to_UIC', 'distance_to_Northwestern', 'distance_to_NEIU',
       'distance_to_Illinois Tech', 'distance_to_DePaul_Loop',
       'distance_to_DePaul_Lincoln_Park', 'distance_to_Loyola_Lake_Shore',
       'distance_to_Loyola_Health_Sciences', 'distance_to_Loyola_Water_Tower',
       'distance_to_Chicago_State']].head()
```

Out[63]:

	GEOID	INTPTLAT	INTPTLON	logPop	logMidHomeValue	logSaleHouseValue	distance_to_UChicago	distance_to_UIC	distance_to_Northwestern	distance_to_NEIU	distance_to_Illinois Tech	distance_to_DePaul_Loop	distance_to_DePaul_Lincoln_Park	distance_to_Loyola_La
0	17031221000	+41.9209604	-087.7152526	7.928046	12.570716	2.302585	22837.922146	9543.405442	19853.038273	8105.724679	16042.463991	11282.666455	6117.381881	127
1	17031221100	+41.9210205	-087.7103609	8.557759	12.746069	3.044522	22456.389068	9034.397115	19706.054739	8124.189957	15626.730254	10675.770660	5391.661358	123
2	17031242800	+41.8926320	-087.6903237	7.340836	13.010090	0.000000	17580.179771	4537.583407	23777.110734	12702.440882	10787.868719	6801.079518	5376.957816	157
3	17031242900	+41.8921955	-087.6842896	7.388328	12.937479	0.000000	17320.705974	4089.832277	23745.884872	12961.290725	10491.517010	6282.694874	5069.011206	156
4	17031243000	+41.8922753	-087.6793981	7.729296	12.892195	2.397895	17054.379038	3653.502848	23726.987406	13116.382128	10195.929179	5761.598949	4803.055148	155

Calculate the distance for each tract to the closest university and mark the corresponding university name in the geodatabase

In [64]:
```
distances = np.array(acs_df_2018[['distance_to_UChicago', 'distance_to_UIC', 'distance_to_Northwestern', 'distance_to_NEIU', 'distance_to_Illinois Tech', 'distance_to_DePaul_Loop', 'distance_to_DePaul_Lincoln_Park', 'distance_to_Loyola_Lake_Shore', 'd
acs_df_2018['dis_to_Uni'] = np.min(distances)
def get_min_distance(row):
    distances = [
        row['distance_to_UChicago'],
        row['distance_to_UIC'],
        row['distance_to_Northwestern'],
        row['distance_to_NEIU'],
        row['distance_to_Illinois Tech'],
        row['distance_to_DePaul_Loop'],
        row['distance_to_DePaul_Lincoln_Park'],
        row['distance_to_Loyola_Lake_Shore'],
        row['distance_to_Loyola_Health_Sciences'],
        row['distance_to_Loyola_Water_Tower'],
        row['distance_to_Chicago_State']
    ]
    return np.min(distances)

acs_df_2018['dis_to_Uni'] = acs_df_2018.apply(get_min_distance, axis=1)
# Create a list of the university distance columns
uni_distance_columns = ['distance_to_UChicago', 'distance_to_UIC', 'distance_to_Northwestern', 'distance_to_NEIU',
                        'distance_to_Illinois Tech', 'distance_to_DePaul_Loop', 'distance_to_DePaul_Lincoln_Park',
                        'distance_to_Loyola_Lake_Shore', 'distance_to_Loyola_Health_Sciences',
                        'distance_to_Loyola_Water_Tower', 'distance_to_Chicago_State']

# Create the 'closestUni' column by finding the column name with the minimum distance
acs_df_2018['closestUni'] = acs_df_2018[uni_distance_columns].idxmin(axis=1)
acs_df_2018['closestUni'] = acs_df_2018['closestUni'].str.replace('distance_to_', '')
```

In [65]:
```
distances = np.array(acs_df_2021[['distance_to_UChicago', 'distance_to_UIC', 'distance_to_Northwestern', 'distance_to_NEIU', 'distance_to_Illinois Tech', 'distance_to_DePaul_Loop', 'distance_to_DePaul_Lincoln_Park', 'distance_to_Loyola_Lake_Shore', 'd
acs_df_2021['dis_to_Uni'] = np.min(distances)
def get_min_distance(row):
    distances = [
        row['distance_to_UChicago'],
        row['distance_to_UIC'],
        row['distance_to_Northwestern'],
        row['distance_to_NEIU'],
        row['distance_to_Illinois Tech'],
        row['distance_to_DePaul_Loop'],
        row['distance_to_DePaul_Lincoln_Park'],
        row['distance_to_Loyola_Lake_Shore'],
        row['distance_to_Loyola_Health_Sciences'],
        row['distance_to_Loyola_Water_Tower'],
        row['distance_to_Chicago_State']
    ]
    return np.min(distances)

acs_df_2021['dis_to_Uni'] = acs_df_2021.apply(get_min_distance, axis=1)
# Create a list of the university distance columns
uni_distance_columns = ['distance_to_UChicago', 'distance_to_UIC', 'distance_to_Northwestern', 'distance_to_NEIU',
                        'distance_to_Illinois Tech', 'distance_to_DePaul_Loop', 'distance_to_DePaul_Lincoln_Park',
                        'distance_to_Loyola_Lake_Shore', 'distance_to_Loyola_Health_Sciences',
                        'distance_to_Loyola_Water_Tower', 'distance_to_Chicago_State']

# Create the 'closestUni' column by finding the column name with the minimum distance
acs_df_2021['closestUni'] = acs_df_2021[uni_distance_columns].idxmin(axis=1)
acs_df_2021['closestUni'] = acs_df_2021['closestUni'].str.replace('distance_to_', '')
```

In [66]:
```
acs_df_2021.head(1)
```

Out[66]:

	STATEFP	COUNTYFP	TRACTCE	GEOID	NAME_x	NAMELSAD	MTFCC	FUNCSTAT	ALAND	AWATER	...	distance_to_NEIU	distance_to_Illinois Tech	distance_to_DePaul_Loop	distance_to_DePaul_Lincoln_Park	distance_to_Loyola_Lake_Shore	distance_to_Loyola_Health_Science
0	17	031	221000	17031221000	2210	Census Tract 2210	G5020	S	220053	0	...	8105.724679	16042.463991	11282.666455	6117.381881	12700.475695	15141.92288

1 rows × 48 columns

Create a dummy variable to identify if the closest university to this tract is a private university

In [67]:
```
private_campuses = ['UChicago', 'Northwestern', 'Illinois Tech', 'Loyola_Lake_Shore', 'Loyola_Health_Sciences', 'Loyola_Water_Tower']
acs_df_2018['is_private'] = acs_df_2018['closestUni'].apply(lambda x: 1 if x in private_campuses else 0)

acs_df_2021['is_private'] = acs_df_2021['closestUni'].apply(lambda x: 1 if x in private_campuses else 0)
acs_df_2021.head(1)
```

Out[67]:

	STATEFP	COUNTYFP	TRACTCE	GEOID	NAME_x	NAMELSAD	MTFCC	FUNCSTAT	ALAND	AWATER	...	distance_to_Illinois Tech	distance_to_DePaul_Loop	distance_to_DePaul_Lincoln_Park	distance_to_Loyola_Lake_Shore	distance_to_Loyola_Health_Sciences	distance_to_Loyc
0	17	031	221000	17031221000	2210	Census Tract 2210	G5020	S	220053	0	...	16042.463991	11282.666455	6117.381881	12700.475695	15141.922884	

1 rows × 49 columns

In [68]:
```
acs_df_2018['dis_to_Uni'].describe()
```

Out[68]:
```
count     1288.000000
mean     10174.824797
std      10380.696481
min          0.000000
25%       3153.867030
50%       6482.344858
75%      12651.090871
max      50935.555851
Name: dis_to_Uni, dtype: float64
```

Calculate the distance to the nearest bus stop and rail station

In [69]:
```
# Reproject cta_bus_stops and cta_rail_stations to a projected CRS
cta_bus_stops = cta_bus_stops.to_crs(epsg=3857)
cta_rail_stations = cta_rail_stations.to_crs(epsg=3857)
from shapely.ops import nearest_points

# Function to calculate the distance to the nearest bus stop
def calculate_distance_to_nearest_stop(row, stops_gdf):
    # Find the nearest bus stop
    nearest_stop = nearest_points(row['geometry'], stops_gdf.unary_union)[1]
    # Calculate the distance
    distance = row['geometry'].distance(nearest_stop)
    return distance

# Apply the function to each row in the acs_df GeoDataFrame
acs_df_2018['dis_to_bus'] = acs_df_2018.apply(calculate_distance_to_nearest_stop, axis=1, stops_gdf=cta_bus_stops)
acs_df_2018['dis_to_rail'] = acs_df_2018.apply(calculate_distance_to_nearest_stop, axis=1, stops_gdf=cta_rail_stations)
acs_df_2021['dis_to_bus'] = acs_df_2021.apply(calculate_distance_to_nearest_stop, axis=1, stops_gdf=cta_bus_stops)
acs_df_2021['dis_to_rail'] = acs_df_2021.apply(calculate_distance_to_nearest_stop, axis=1, stops_gdf=cta_rail_stations)
```

Merge CTA Rail Ridership

In [70]:
```
# Function to calculate the distance to the nearest rail station and get its ID and ridership
def calculate_distance_and_ridership(row, stations_gdf, ridership_gdf):
    # Find the nearest rail station
    nearest_station = nearest_points(row['geometry'], stations_gdf.unary_union)[1]
    # Get the station ID
    station_id = stations_gdf[stations_gdf['geometry'] == nearest_station]['STATION_ID'].values[0]
    # Get the ridership for the corresponding station ID
    ridership_2018 = ridership_gdf[ridership_gdf['STATION_ID'] == station_id]['RIDE2018'].values[0]
    ridership_2021 = ridership_gdf[ridership_gdf['STATION_ID'] == station_id]['RIDE2021'].values[0]
    # Calculate the distance
    distance = row['geometry'].distance(nearest_station)
    return pd.Series([distance, station_id, ridership_2018, ridership_2021], index=['dis_to_rail', 'station_id', 'ridership_2018', 'ridership_2021'])

# Apply the function to each row in the acs_df GeoDataFrame
acs_df_2018[['dis_to_rail', 'station_id', 'ridership_2018', 'ridership_2021']] = acs_df_2018.apply(calculate_distance_and_ridership, axis=1, stations_gdf=cta_rail_stations, ridership_gdf=cta_rail_ridership)
acs_df_2021[['dis_to_rail', 'station_id', 'ridership_2018', 'ridership_2021']] = acs_df_2021.apply(calculate_distance_and_ridership, axis=1, stations_gdf=cta_rail_stations, ridership_gdf=cta_rail_ridership)
```

Determine if the tract is CBD and calculate the distance for each tract to CBD

In [71]:
```
# Load the Central Business District GeoJSON file
cbd_geojson = gpd.read_file("data\Central_Business_District_20240517.geojson")

# Ensure both GeoDataFrames use the same CRS
cbd_geojson = cbd_geojson.to_crs(acs_df_2018.crs)
# Calculate the centroid of the CBD if it's a polygon
cbd_centroid = cbd_geojson.geometry.centroid.iloc[0]
# Calculate the distance from each tract to the CBD centroid
acs_df_2018['dis_to_CBD'] = acs_df_2018.geometry.distance(cbd_centroid)
# Dummy 'is_CBD'
joined_df = gpd.sjoin(acs_df_2018, cbd_geojson, how="left", op="intersects")
joined_df['is_CBD'] = joined_df['index_right'].apply(lambda x: 1 if pd.notnull(x) else 0)
acs_df_2018 = acs_df_2018.merge(joined_df[['GEOID', 'is_CBD']], on='GEOID', how='left')

#2021
acs_df_2021['dis_to_CBD'] = acs_df_2021.geometry.distance(cbd_centroid)
# Dummy 'is_CBD'
joined_df = gpd.sjoin(acs_df_2021, cbd_geojson, how="left", op="intersects")
joined_df['is_CBD'] = joined_df['index_right'].apply(lambda x: 1 if pd.notnull(x) else 0)
acs_df_2021 = acs_df_2021.merge(joined_df[['GEOID', 'is_CBD']], on='GEOID', how='left')
```

In [72]:
```
acs_df_2021.head(1)
```

Out[72]:

	STATEFP	COUNTYFP	TRACTCE	GEOID	NAME_x	NAMELSAD	MTFCC	FUNCSTAT	ALAND	AWATER	...	dis_to_Uni	closestUni	is_private	dis_to_bus	dis_to_rail	station_id	ridership_2018	ridership_2021	dis_to_CBD	is_CBD
0	17	031	221000	17031221000	2210	Census Tract 2210	G5020	S	220053	0	...	6117.381881	DePaul_Lincoln_Park	0	0.0	956.184241	1020.0	7484.916667	2239.25	10587.762563	0

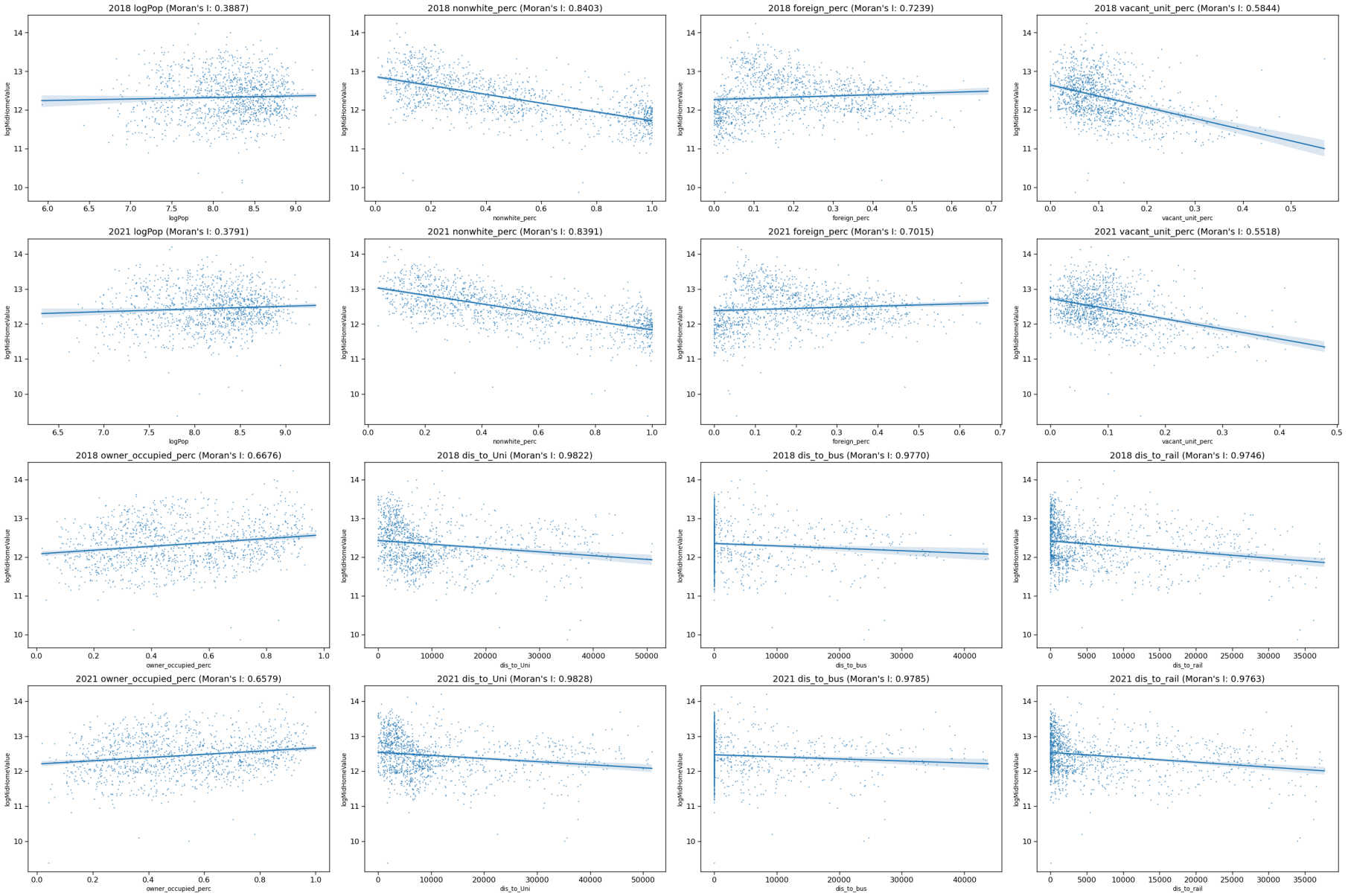
1 rows × 56 columns

6.3 Global Moran's I

6.3.1 Calculation for Global Moran's I

In [103…
```
from esda.moran import Moran
import libpysal

# Assuming acs_df_2018 and acs_df_2021 are your GeoDataFrames for the years 2018 and 2021

# List of columns to include in the Moran's plot matrix
columns = [
    'logPop', 'nonwhite_perc', 'foreign_perc', 'vacant_unit_perc',
    'owner_occupied_perc', 'dis_to_Uni', 'dis_to_bus', 'dis_to_rail'
]

# Function to create a spatial weights matrix
def create_weights(df):
    return libpysal.weights.Queen.from_dataframe(df)

# Function to calculate Moran's I for all columns
def calculate_all_morans_i(df, columns, w):
    return {column: Moran(df[column], w).I for column in columns}

# Create spatial weights matrices for both years
w_2018 = create_weights(acs_df_2018)
w_2021 = create_weights(acs_df_2021)

# Calculate Moran's I for all columns for both years
morans_i_2018 = calculate_all_morans_i(acs_df_2018, columns, w_2018)
morans_i_2021 = calculate_all_morans_i(acs_df_2021, columns, w_2021)

# Create the subplot grid with 4x4
fig, axs = plt.subplots(4, 4, figsize=(30, 20))  # Adjusted the size for better visibility

# Loop to plot each variable
for i, col in enumerate(columns):
    row = i // 4
    col_idx = i % 4
    sns.regplot(x=col, y='logMidHomeValue', data=acs_df_2018, ax=axs[row*2, col_idx], scatter_kws={'s': 0.5})
    axs[row*2, col_idx].set_title(f"2018 {col} (Moran's I: {morans_i_2018[col]:.4f})", fontsize=14)
    axs[row*2, col_idx].tick_params(axis='both', which='major', labelsize=12)

    sns.regplot(x=col, y='logMidHomeValue', data=acs_df_2021, ax=axs[row*2 + 1, col_idx], scatter_kws={'s': 0.5})
    axs[row*2 + 1, col_idx].set_title(f"2021 {col} (Moran's I: {morans_i_2021[col]:.4f})", fontsize=14)
    axs[row*2 + 1, col_idx].tick_params(axis='both', which='major', labelsize=12)

# Adjust layout and add a figure-wide title
plt.tight_layout()
```

```
plt.show()
```



6.3.2 Interpretation of Global Moran's I Graphs

These two sets of graphs present the Global Moran's I analysis results for variables such as population logarithms, percentage of non-white people, percentage of foreign population, percentage of vacant units, percentage of homeowners, and distance to universities, buses, and railway stations in 2018 and 2021. These results reveal the spatial autocorrelation of these variables in the Chicago area, with positive values indicating a tendency for features to cluster, especially for proximity to universities and transportation facilities. From 2018 to 2021, the clustering of non white and foreign populations has increased, while the clustering of vacant units has slightly decreased.

Variables such as distance to universities, bus, and rail stations show very high spatial clustering, suggesting that properties close to these facilities tend to geographically cluster. This generally means that areas close to transit facilities may command higher real estate values due to the added convenience, which is considered a value enhancer. From 2018 to 2021, the clustering near these transit areas has increased, likely reflecting a rising demand for quick urban commutes, thereby driving up real estate values in these areas. The increased spatial clustering of non-white and foreign-born populations might indicate that certain neighborhoods are becoming more homogenous in terms of race and culture. This enhanced clustering can be associated with segmentation in the housing market or with rising prices in certain areas, as they may become more desirable to specific ethnic groups. This also may point to socio-economic stratification in urban areas leading to market segmentation.

6.4 OLS Regression

6.4.1 OLS for 2018

In [73]:
```
variable_names = ['nonwhite_perc', 'foreign_perc', 'vacant_unit_perc', 'dis_to_rail']
```

In [74]:
```
Varlist18= variable_names + ["ridership_2018"]
acs_df_2018[acs_df_2018[Varlist18].isnull().any(axis=1)]
```

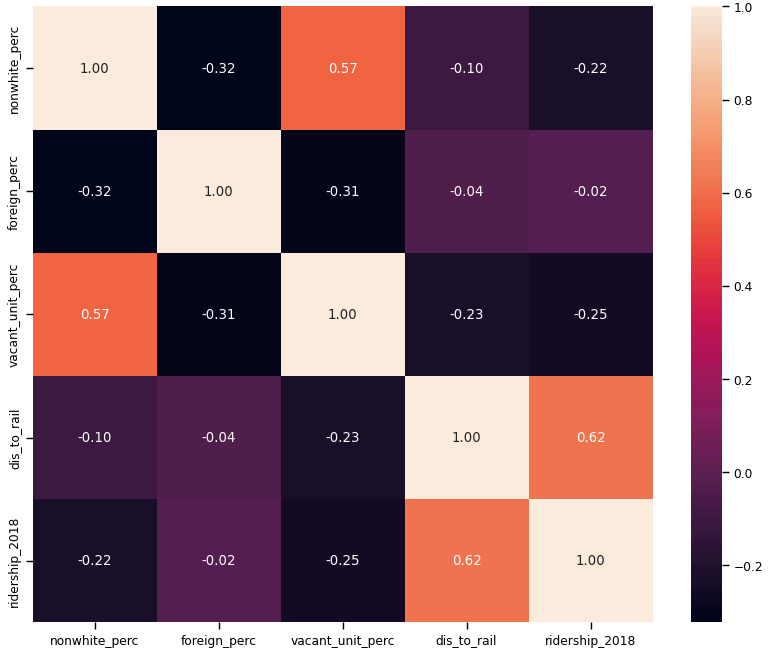
Out[74]:

	STATEFP	COUNTYFP	TRACTCE	GEOID	NAME_x	NAMELSAD	MTFCC	FUNCSTAT	ALAND	AWATER	...	dis_to_Uni	closestUni	is_private	dis_to_bus	dis_to_rail	station_id	ridership_2018	ridership_2021	dis_to_CBD	is_CBD
55	17	031	980100	17031980100	9801	Census Tract 9801	G5020	S	2981781	0	...	13402.991508	Loyola_Health_Sciences	1	0.000000	0.000000	930.0	8599.833333	3180.250000	18247.821379	0
323	17	031	980000	17031980000	9800	Census Tract 9800	G5020	S	19890200	92402	...	14399.445316	Loyola_Health_Sciences	1	3325.041188	0.000000	890.0	11400.916667	5028.833333	30080.554490	0
750	17	031	381700	17031381700	3817	Census Tract 3817	G5020	S	196802	0	...	3311.314365	Illinois Tech	1	0.000000	292.471936	1230.0	2879.500000	1110.916667	11248.677147	0
1161	17	031	990000	17031990000	9900	Census Tract 9900	G5020	S	0	1717072182	...	755.516274	Loyola_Lake_Shore	1	296.765334	1028.539570	1300.0	5129.833333	1953.583333	2233.874727	1

4 rows × 56 columns

In [75]:
```
acs_df_2018 = acs_df_2018.dropna(subset=Varlist18+["logMidHomeValue"])
```

In [76]:
```
correlation_matrix = acs_df_2018[Varlist18].corr()
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, fmt=".2f")
plt.show()
```



In [77]:
```
from statsmodels.stats.outliers_influence import variance_inflation_factor
vif_data = pd.DataFrame()
vif_data["feature"] = acs_df_2018[Varlist18].columns
vif_data["VIF"] = [variance_inflation_factor(acs_df_2018[Varlist18].values, i) for i in range(len(acs_df_2018[Varlist18].columns))]
vif_data
```

Out[77]:

	feature	VIF
0	nonwhite_perc	4.199090
1	foreign_perc	1.908475
2	vacant_unit_perc	4.076822
3	dis_to_rail	2.522510
4	ridership_2018	4.002518

In [78]:
```
OLS18 = spreg.OLS(
    acs_df_2018[["logMidHomeValue"]].values,
    acs_df_2018[Varlist18].values,
    name_y="logMidHomeValue",
    name_x=Varlist18,
)
print(OLS18.summary)
```

```
REGRESSION RESULTS
------------------

SUMMARY OF OUTPUT: ORDINARY LEAST SQUARES
-----------------------------------------
Data set            :     unknown
Weights matrix      :        None
Dependent Variable  :logMidHomeValue               Number of Observations:        1273
Mean dependent var  :     12.3303               Number of Variables   :           6
S.D. dependent var  :      0.5604               Degrees of Freedom    :        1267
R-squared           :      0.5475
Adjusted R-squared  :      0.5457
Sum squared residual:     180.722               F-statistic           :    306.6374
Sigma-square        :       0.143               Prob(F-statistic)     :  3.197e-215
S.E. of regression  :       0.378               Log likelihood        :    -563.753
Sigma-square ML     :       0.142               Akaike info criterion :    1139.505
S.E of regression ML:      0.3768               Schwarz criterion     :    1170.400

------------------------------------------------------------------------------------
            Variable     Coefficient       Std.Error     t-Statistic     Probability
------------------------------------------------------------------------------------
            CONSTANT        13.40960         0.03815       351.51406         0.00000
       nonwhite_perc        -1.16660         0.04251       -27.44573         0.00000
         foreign_perc        -0.78300         0.07915        -9.89314         0.00000
    vacant_unit_perc        -1.38949         0.17113        -8.11926         0.00000
         dis_to_rail        -0.00002         0.00000        -9.92575         0.00000
      ridership_2018        -0.00003         0.00000        -6.67313         0.00000
------------------------------------------------------------------------------------

REGRESSION DIAGNOSTICS
MULTICOLLINEARITY CONDITION NUMBER            8.662

TEST ON NORMALITY OF ERRORS
TEST                             DF        VALUE           PROB
Jarque-Bera                       2         225.992           0.0000

DIAGNOSTICS FOR HETEROSKEDASTICITY
RANDOM COEFFICIENTS
TEST                             DF        VALUE           PROB
Breusch-Pagan test                5         134.065           0.0000
Koenker-Bassett test              5          66.112           0.0000
================================ END OF REPORT =====================================
```

6.4.2 OLS Model for 2021

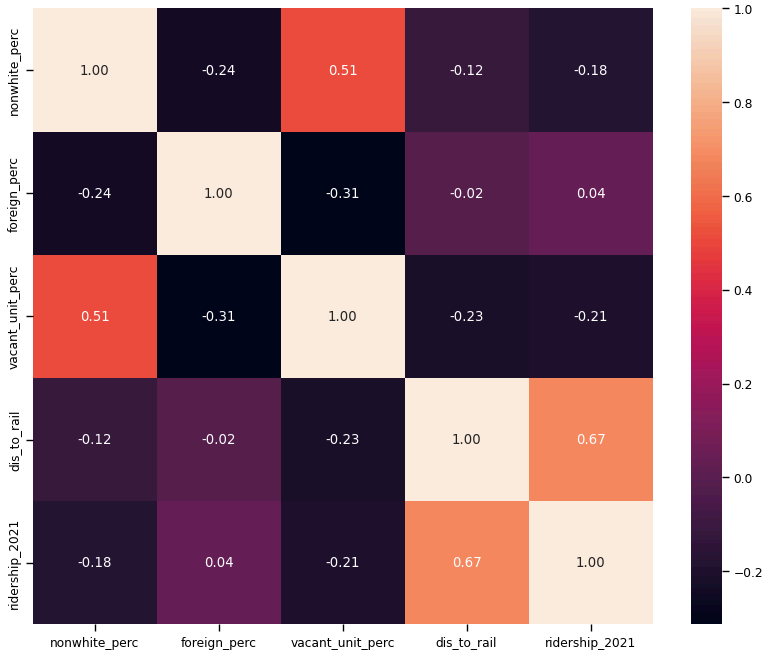
In [79]:
```
Varlist21= variable_names + ["ridership_2021"]
acs_df_2021 = acs_df_2021.dropna(subset=Varlist21+["logMidHomeValue"])
acs_df_2021[acs_df_2021[Varlist21].isnull().any(axis=1)]
```

Out[79]:

STATEFP	COUNTYFP	TRACTCE	GEOID	NAME_x	NAMELSAD	MTFCC	FUNCSTAT	ALAND	AWATER	...	dis_to_Uni	closestUni	is_private	dis_to_bus	dis_to_rail	station_id	ridership_2018	ridership_2021	dis_to_CBD	is_CBD

0 rows × 56 columns

In [80]:
```
correlation_matrix = acs_df_2021[Varlist21].corr()
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, fmt=".2f")
plt.show()
```



In [81]:
```
vif_data["feature"] = acs_df_2021[Varlist21].columns
vif_data["VIF"] = [variance_inflation_factor(acs_df_2021[Varlist21].values, i) for i in range(len(acs_df_2021[Varlist21].columns))]
vif_data
```

Out[81]:

	feature	VIF
0	nonwhite_perc	4.442830
1	foreign_perc	2.061458
2	vacant_unit_perc	3.761674
3	dis_to_rail	2.854869
4	ridership_2021	4.462365

In [82]:
```
OLS21 = spreg.OLS(
    acs_df_2021[["logMidHomeValue"]].values,
    acs_df_2021[Varlist21].values,
    name_y="logMidHomeValue",
    name_x=Varlist21,
)
print(OLS21.summary)
```

```
REGRESSION RESULTS
------------------

SUMMARY OF OUTPUT: ORDINARY LEAST SQUARES
-----------------------------------------
Data set            :     unknown
Weights matrix      :        None
Dependent Variable  :logMidHomeValue               Number of Observations:        1312
Mean dependent var  :     12.4445               Number of Variables   :           6
S.D. dependent var  :      0.5577               Degrees of Freedom    :        1306
R-squared           :      0.5546
Adjusted R-squared  :      0.5529
Sum squared residual:     181.573               F-statistic           :    325.2886
Sigma-square        :       0.139               Prob(F-statistic)     :  2.122e-226
S.E. of regression  :       0.373               Log likelihood        :    -564.309
Sigma-square ML     :       0.138               Akaike info criterion :    1140.619
S.E of regression ML:      0.3720               Schwarz criterion     :    1171.695

------------------------------------------------------------------------------------
            Variable     Coefficient       Std.Error     t-Statistic     Probability
------------------------------------------------------------------------------------
            CONSTANT        13.51249         0.03564       379.17654         0.00000
       nonwhite_perc        -1.25649         0.04149       -30.28275         0.00000
         foreign_perc        -0.47560         0.07738        -6.14611         0.00000
    vacant_unit_perc        -1.19184         0.17808        -6.69273         0.00000
         dis_to_rail        -0.00001         0.00000        -8.75023         0.00000
       ridership_2021        -0.00006         0.00001        -6.96717         0.00000
------------------------------------------------------------------------------------

REGRESSION DIAGNOSTICS
MULTICOLLINEARITY CONDITION NUMBER            8.319

TEST ON NORMALITY OF ERRORS
TEST                             DF        VALUE           PROB
Jarque-Bera                       2         671.549           0.0000

DIAGNOSTICS FOR HETEROSKEDASTICITY
RANDOM COEFFICIENTS
TEST                             DF        VALUE           PROB
Breusch-Pagan test                5          68.764           0.0000
Koenker-Bassett test              5          25.558           0.0001
================================= END OF REPORT =====================================
```

6.4.3 Interpretation of OLS Models

According to the OLS model in 2018, the negative coefficients for demographic factors (nonwhite_perc, foreign_perc) and vacancy rates may suggest socio-economic factors or preferences affecting housing values in areas with higher proportions of these demographics or more vacant homes. The slight influence of distance to rail stations on lowering home values could indicate a preference for proximity to transit options, which typically enhances property values. However, the negative effect of ridership may need further exploration to understand underlying causes, such as whether higher ridership is associated with negative externalities that affect housing prices. This may indicates that the affluent group does not tend to use public transit.

In 2021, all predictors are highly significant with p-values of 0.00000, similar to the 2018 model, suggesting strong evidence against the null hypothesis for all included variables. In the 2021 model, notably, the impact of the non-white percentage has become more negative, and the coefficient for ridership is more negative compared to the previous year. This suggests that an increase in ridership now has a stronger negative correlation with housing values than it did before. This shows that in the post- Covid 19 era, the difference in transportation usage between ethnic minorities and white people is increasing, and the use of public transportation and subway distances from properties is more associated with lower housing affordability.

6.5 Spatial Regime Model

6.5.1 Spatial Regime Model for 2018

In [83]:
```
SR18 = spreg.OLS_Regimes(
    acs_df_2018[["logMidHomeValue"]].values,
    acs_df_2018[Varlist18].values,
    acs_df_2018["is_CBD"].tolist(),
    constant_regi="many",
    regime_err_sep=False,
    name_y="logMidHomeValue",
    name_x=Varlist18,
)
```

In [84]:
```
res = pd.DataFrame(
    {
        # Pull out regression coefficients and
        # flatten as they are returned as Nx1 array
        "Coeff.": SR18.betas.flatten(),
        # Pull out and flatten standard errors
        "Std. Error": SR18.std_err.flatten(),
        # Pull out P-values from t-stat object
        "P-Value": [i[1] for i in SR18.t_stat],
    },
    index=SR18.name_x,
)
```

```
## Extract variables for the coastal regime
CBD = [i for i in res.index if "1_" in i]
CBD = res.loc[CBD, :].rename(lambda i: i.replace("1_", ""))
## Build multi-index column names
CBD.columns = pd.MultiIndex.from_product(
    [["CBD"], CBD.columns]
)
# Non-CBD
## Extract variables for the non-coastal regime
out = [i for i in res.index if "0_" in i]
out = res.loc[out, :].rename(lambda i: i.replace("0_", ""))
## Build multi-index column names
out.columns = pd.MultiIndex.from_product(
    [["Outside"], out.columns]
)
# Concat both models
pd.concat([CBD, out], axis=1)
```

Out[84]:

	CBD			Outside		
	Coeff.	Std. Error	P-Value	Coeff.	Std. Error	P-Value
CONSTANT	12.972485	0.269899	2.508697e-287	13.455864	0.037573	0.000000e+00
nonwhite_perc	0.028624	0.843532	9.729353e-01	-1.100378	0.041823	5.689917e-122
foreign_perc	-1.023915	1.081883	3.441164e-01	-0.812831	0.076990	5.009319e-25
vacant_unit_perc	0.618738	0.838265	4.605805e-01	-1.851549	0.173497	1.627757e-25
dis_to_rail	0.000271	0.000317	3.934378e-01	-0.000013	0.000002	6.172307e-16
ridership_2018	0.000012	0.000023	6.145541e-01	-0.000035	0.000004	3.158383e-19

6.5.2 Spatial Regime Model for 2021

In [85]:
```
SR21 = spreg.OLS_Regimes(
    acs_df_2021[["logMidHomeValue"]].values,
    acs_df_2021[Varlist21].values,
    acs_df_2021["is_CBD"].tolist(),
    constant_regi="many",
    regime_err_sep=False,
    name_y="logMidHomeValue",
    name_x=Varlist21,
)
```

In [86]:
```
res = pd.DataFrame(
    {
        # Pull out regression coefficients and
        # flatten as they are returned as Nx1 array
        "Coeff.": SR21.betas.flatten(),
        # Pull out and flatten standard errors
        "Std. Error": SR21.std_err.flatten(),
        # Pull out P-values from t-stat object
        "P-Value": [i[1] for i in SR21.t_stat],
    },
    index=SR21.name_x,
)
## Extract variables for the coastal regime
CBD = [i for i in res.index if "1_" in i]
CBD = res.loc[CBD, :].rename(lambda i: i.replace("1_", ""))
## Build multi-index column names
CBD.columns = pd.MultiIndex.from_product(
    [["CBD"], CBD.columns]
)
# Non-CBD
## Extract variables for the non-coastal regime
out = [i for i in res.index if "0_" in i]
out = res.loc[out, :].rename(lambda i: i.replace("0_", ""))
## Build multi-index column names
out.columns = pd.MultiIndex.from_product(
    [["Outside"], out.columns]
)
# Concat both models
pd.concat([CBD, out], axis=1)
```

Out[86]:

	CBD			Outside		
	Coeff.	Std. Error	P-Value	Coeff.	Std. Error	P-Value
CONSTANT	13.266159	0.303604	2.258006e-257	13.528267	0.035486	0.000000e+00
nonwhite_perc	-0.906958	0.787932	2.499188e-01	-1.201198	0.041690	1.383110e-141
foreign_perc	0.110858	0.759801	8.840197e-01	-0.511979	0.076788	3.837076e-11
vacant_unit_perc	0.715346	0.988215	4.692716e-01	-1.525842	0.182153	1.393752e-16
dis_to_rail	0.000205	0.000305	5.016357e-01	-0.000012	0.000002	1.243038e-13
ridership_2021	-0.000029	0.000064	6.472906e-01	-0.000078	0.000009	2.536415e-16

6.5.3 Interpretation for Spatial Regime Model

The spatial regime models for 2018 and 2021 reveal distinct impacts of various factors on home values within the Central Business District (CBD) versus outside the CBD. In 2018, variables like the percentage of non-white residents, foreign-born residents, and proximity to rail had minimal influence on home values within the CBD but were significant predictors outside the CBD, indicating demographic and transit proximity factors more strongly impact peripheral areas. By 2021, this trend continued, with non-white percentage now also significantly affecting the CBD, suggesting changing urban dynamics. Both years show that variables related to demographics and transit continue to negatively impact home values outside the CBD much more than in the CBD area. The change in ridership in the CBD area from positive to negative indicates that people in the CBD may use public transportation less, which may be due to more remote office work after the epidemic. The impact of ridership in areas outside of CBD on housing prices has always been negative, indicating that low-income suburban communities still rely more on public transportation for commuting.

6.6 Spatial Lag Model

6.6.1 Spatial Lag Model for 2018

In [87]:
```
knn18 = weights.KNN.from_dataframe(acs_df_2018, k=20)
SLM18 = spreg.GM_Lag(
    acs_df_2018[["logMidHomeValue"]].values,
    acs_df_2018[Varlist18].values,
    w=knn18,
    name_y="logMidHomeValue",
    name_x=Varlist18,
)
print(SLM18.summary)
```

```
REGRESSION RESULTS
------------------

SUMMARY OF OUTPUT: SPATIAL TWO STAGE LEAST SQUARES
--------------------------------------------------
Data set            :     unknown
Weights matrix      :     unknown
Dependent Variable  :logMidHomeValue                Number of Observations:        1273
Mean dependent var  :     12.3303               Number of Variables   :           7
S.D. dependent var  :      0.5604               Degrees of Freedom    :        1266
Pseudo R-squared    :      0.7732
Spatial Pseudo R-squared:  0.6118

------------------------------------------------------------------------------------
            Variable     Coefficient       Std.Error     z-Statistic     Probability
------------------------------------------------------------------------------------
            CONSTANT         4.31300         0.51284         8.41001         0.00000
       nonwhite_perc        -0.49124         0.04854       -10.12075         0.00000
         foreign_perc        -0.61779         0.05694       -10.84917         0.00000
    vacant_unit_perc        -0.81194         0.12575        -6.45677         0.00000
         dis_to_rail        -0.00000         0.00000        -3.70018         0.00022
      ridership_2018        -0.00001         0.00000        -4.50443         0.00001
   W_logMidHomeValue         0.03468         0.00195        17.76240         0.00000
------------------------------------------------------------------------------------
Instrumented: W_logMidHomeValue
Instruments: W_dis_to_rail, W_foreign_perc, W_nonwhite_perc,
             W_ridership_2018, W_vacant_unit_perc
================================ END OF REPORT =====================================
```

6.6.2 Spatial Lag Model for 2021

In [88]:
```
knn21 = weights.KNN.from_dataframe(acs_df_2021, k=20)
SLM21 = spreg.GM_Lag(
    acs_df_2021[["logMidHomeValue"]].values,
    acs_df_2021[Varlist21].values,
    w=knn21,
    name_y="logMidHomeValue",
    name_x=Varlist21,
)
print(SLM21.summary)
```

```
REGRESSION RESULTS
------------------

SUMMARY OF OUTPUT: SPATIAL TWO STAGE LEAST SQUARES
--------------------------------------------------
Data set            :     unknown
Weights matrix      :     unknown
Dependent Variable  :logMidHomeValue               Number of Observations:        1312
Mean dependent var  :     12.4445               Number of Variables   :           7
S.D. dependent var  :      0.5577               Degrees of Freedom    :        1305
Pseudo R-squared    :      0.7552
Spatial Pseudo R-squared:  0.6030

------------------------------------------------------------------------------------
            Variable     Coefficient       Std.Error     z-Statistic     Probability
------------------------------------------------------------------------------------
            CONSTANT         4.45694         0.56793         7.84765         0.00000
       nonwhite_perc        -0.52914         0.05501        -9.61847         0.00000
        foreign_perc        -0.48426         0.05749        -8.42402         0.00000
    vacant_unit_perc        -0.76462         0.13497        -5.66532         0.00000
         dis_to_rail        -0.00000         0.00000        -3.05028         0.00229
       ridership_2021        -0.00003         0.00001        -4.08048         0.00004
   W_logMidHomeValue         0.03421         0.00214        15.96210         0.00000
------------------------------------------------------------------------------------
Instrumented: W_logMidHomeValue
Instruments: W_dis_to_rail, W_foreign_perc, W_nonwhite_perc,
             W_ridership_2021, W_vacant_unit_perc
================================ END OF REPORT =====================================
```

6.6.3 Interpretation for Spatial Lag Model

Spatial Lag of Log Mid Home Values (W_logMidHomeValue) has a positive coefficient in both years, demonstrating the influence of neighboring values on a given location's home value. The constant influence of spatial dependency (W_logMidHomeValue) in both years underscores the need for real estate models to consider spatial interactions to accurately assess property value determinants. This coefficient remains consistently significant, highlighting the importance of spatial effects in housing market analysis. All the indicators showed a negative impact. The increase in the negative impact of demographic factors and ridership over time might suggest that the difference in public transit usage between different groups is enlarging.

6.6 Spatial Durbin Model

6.6.1 Spatial Durbin Model for 2018

In [89]:
```
knn18.transform = "R"
lag_variables_used = acs_df_2018.filter(['nonwhite_perc', 'foreign_perc', 'vacant_unit_perc', 'dis_to_rail', 'ridership_2018'])
wx = lag_variables_used.apply(lambda y: weights.spatial_lag.lag_spatial(knn18, y))
wx = wx.rename(columns=lambda c: "w_"+ c)
slx_exog18 = acs_df_2018[Varlist18].join(wx)
```

In [90]:
```
knn21.transform = "R"
lag_variables_used = acs_df_2021.filter(['nonwhite_perc', 'foreign_perc', 'vacant_unit_perc', 'dis_to_rail', 'ridership_2021'])
wx = lag_variables_used.apply(lambda y: weights.spatial_lag.lag_spatial(knn21, y))
wx = wx.rename(columns=lambda c: "w_"+ c)
slx_exog21 = acs_df_2021[Varlist21].join(wx)
```

In [91]:
```
SDM18 = spreg.GM_Lag(
    acs_df_2018[["logMidHomeValue"]].values,
    slx_exog18.values,
    w=knn18,
    name_y="logMidHomeValue",
    name_x=slx_exog18.columns.tolist(),
)
print(SDM18.summary)
```

```
REGRESSION RESULTS
------------------

SUMMARY OF OUTPUT: SPATIAL TWO STAGE LEAST SQUARES
--------------------------------------------------
Data set            :     unknown
Weights matrix      :     unknown
Dependent Variable  :logMidHomeValue               Number of Observations:        1273
Mean dependent var  :     12.3303               Number of Variables   :          12
S.D. dependent var  :      0.5604               Degrees of Freedom    :        1261
Pseudo R-squared    :      0.0202
Spatial Pseudo R-squared: omitted due to rho outside the boundary (-1, 1).

------------------------------------------------------------------------------------
            Variable     Coefficient       Std.Error     z-Statistic     Probability
------------------------------------------------------------------------------------
            CONSTANT       -34.42374
       nonwhite_perc        -2.78033         5.08284        -0.54700         0.58438
        foreign_perc         2.70351         8.60099         0.31433         0.75327
    vacant_unit_perc        -2.36103        13.75619        -0.17163         0.86373
         dis_to_rail         0.00001         0.00063         0.01937         0.98454
       ridership_2018         0.00008         0.00052         0.14965         0.88104
     w_nonwhite_perc         6.50409
      w_foreign_perc         1.11162        14.24899         0.07801         0.93782
  w_vacant_unit_perc         7.89387
       w_dis_to_rail        -0.00002         0.00007        -0.23264         0.81604
     w_ridership_2018         0.00001         0.00020         0.03967         0.96835
   W_logMidHomeValue         1.45985
------------------------------------------------------------------------------------
Instrumented: W_logMidHomeValue
Instruments: W_dis_to_rail, W_foreign_perc, W_nonwhite_perc,
             W_ridership_2018, W_vacant_unit_perc, W_w_dis_to_rail,
             W_w_foreign_perc, W_w_nonwhite_perc, W_w_ridership_2018,
             W_w_vacant_unit_perc
Warning: *** WARNING: Estimate for spatial lag coefficient is outside the boundary (-1, 1). ***
================================ END OF REPORT =====================================
```

6.6.2 Limitation for Spatial Durbin Model

When comparing the Spatial Durbin Model (SDM) and Spatial Lag Model (SLM), SDM shows significantly lower pseudo R-squared values than SLM, indicating weaker explanatory variable variability. In addition, the estimated value of the spatial lag coefficient of SDM exceeds the theoretical boundary, indicating that the model may not be suitable for the data structure. Therefore, it has been decided not to continue using SDM and instead adopt the better performing SLM.

7 Limitations

Overall, while the results of the study can explain the research questions, they did not meet the initial expectations. We started off with a very large number of ideas when exploring the dataset, which can be found in the pre-processing of the data, where we calculated the ridership of each bus route on weekdays, Saturdays, and Sundays, and we also planned to use whether or not the subway stations are equipped with ADA accessibility features and parking lots as dummy variables, and we also planned to explore whether or not the overnight subway is also a factor that affects the price of the house (the red line and the blue line). We have listed the various characteristics of transportation that may affect house prices and processed the data accordingly in our preliminary exploration, and have devoted a lot of time to this.

As the research progressed, a variety of problems arrived, starting with the fact that many of the calculated variables did not work well. For example, the passenger volume of bus routes could not be accurately assigned to individual stops, and could not be combined with the census tract for spatial analysis. We also computed a number of distance variables for universities to examine whether transportation around universities has a greater impact on housing prices, since we assume that college students in the city of Chicago commute predominantly by public transportation. However, at the end of the regression analysis, the entire calculation was abandoned because the VIFs of the variables were too large.

Problems within the dataset were also reflected in the regression analysis. Mathematical problems, such as that most tracts contained bus stops or are adjacent to them, resulted in more than half of the tracts having a 0 distance to a bus stop, so none of the regression models were successful in calculating the weights for that variable. Similarly, there are cases where log values are negative, weight matrices are singular, and so on. Unlike syntax errors, these are only possible causes of the error that can be found on the web, with no specific solution, so sorting out the data also becomes one of the factors limiting the depth of the research beyond the availability of the data.

8 Conclusion

This research explores the intricate relationship between housing prices, public transportation accessibility, and urban development in Chicago during 2018 and 2021. It delves into how proximity to transit options—such as subways and buses—and the demographic shifts within Chicago influence the housing market. The analysis, underpinned by spatial models and Ordinary Least Squares (OLS) regression, reveals significant spatial autocorrelation and the varying impacts of demographic and economic factors on housing values across different urban areas.

The findings highlight that areas near transit facilities tend to maintain higher property values due to the convenience provided, with increased demand for quick commutes notably driving up prices from 2018 to 2021. Simultaneously, the increased spatial clustering of non-white and foreign-born populations indicates growing homogeneity in certain neighborhoods, which might lead to market segmentation based on racial and cultural lines. Moreover, the data suggest that demographic factors and the proximity to transit significantly affect housing values outside the Central Business District more than within it, implying that peripheral areas are more sensitive to these variables.

Additionally, the study observes a paradox where housing prices are higher near transit facilities, yet there is a negative correlation between ridership and housing prices, suggesting a mismatch of transportation resources. This indicates that while transit accessibility increases property values, the benefits of this accessibility may not be equitably distributed among all residents, particularly those who rely more heavily on public transit. This misalignment calls for urban planning that increases equity, ensuring that enhancements in transit infrastructure benefit all segments of the urban population.

These insights are critical for policymakers and urban planners aiming to foster equitable urban growth and enhance the livability of urban spaces while ensuring efficient access to essential services and maintaining housing affordability. The research underscores the need for nuanced urban planning strategies that consider the diverse needs of urban populations and the dynamic nature of urban transit systems.

9 Reference

Brueckner, J. K., & Rosenthal, S. S. (2009). Gentrification and neighborhood housing cycles: will America's future downtowns be rich?. The Review of Economics and Statistics, 91(4), 725-743.

Crum, G. (2020). Impact of Public Transit Access on Chicago Housing Prices.

Herath, S., & Maier, G. (2010). The hedonic price method in real estate and housing market research: a review of the literature.

Hess, D. B., & Almeida, T. M. (2007). Impact of proximity to light rail rapid transit on station-area property values in Buffalo, New York. Urban Studies, 44(5-6), 1041-1068.

Lee, S., & Smart, M. (2020). Public Transportation Accessibility, Proximity to New York City, and Residential Property Values in New Jersey. Hatfield Graduate Journal of Public Affairs, 4(1), 6.

Liu, D., Kwan, M. P., Kan, Z., & Song, Y. (2021). An integrated analysis of housing and transit affordability in the Chicago metropolitan area. The Geographical Journal, 187(2), 110-126.

Giuliano, G. (2005). Low income, public transit, and mobility. Transportation Research Record, 1927(1), 63-70.

Pamidimukkala, R. (2016). Impact of Rapid Transit on the Residential Market in Hudson County, New Jersey.

Regional Transportation Authority. (2021, April 19). Survey shows current riders are more likely to be essential workers or minorities, feel transit is safe. Retrieved from https://www.rtachicago.org/blog/2021/04/19/survey-shows-current-riders-are-more-likely-to-be-essential-workers-or-minorities-feel-transit-is-safe#:~:text=Survey%20results%20show%20that%20among,and%2012%20percent%20are%20Latino.

TransitCenter. (2021). TransitCenter Equity Dashboard: The Chicago Story. Retrieved from https://dashboard.transitcenter.org/story/chicago

U.S. Census Bureau. (2018). Commuting Characteristics by Sex: 5-year estimates, American Community Survey. Retrieved from https://data.ce nsus.gov/cedsci/table?t=Commuting&g=0100000US&y=2018&tid=ACSST5Y2018.S0801&hidePreview=false