What drives commuting decisions?

2601547

2025-05-02

Abstract

This study answers, “which structural socio-economic factors between regions most influence the relative popularity of commuting routes?” This question is important as transport policy, development planning and wider government initiatives such as the “Levelling Up” agenda are all influenced by commuting patterns. The analysis uses a dataset of commuting flows between local authorities in England. It employs statistical and machine learning techniques (such as OLS, SEM and Random Forest) to identify the most significant factors influencing commuting decisions. The study finds that across all methods, distance is the most important factor influencing commuting flows, generally followed by population, median pay and housing growth differences between local authorities. The study also finds that the relative importance of these factors varies by region, with spatial aspects playing a key role in the influence of different factors.

# 1. Research Question

The question asks, “what factors are influential in driving commuting decisions?” The provided dataset includes daily commuting data between local authorities (LAs) in England over the course of 2019. 'journey\_score,’ the key candidate for the dependent variable, represents the popularity of each route.

As with all inferential tasks, there are multiple paths to understanding and addressing the question.  The dataset is daily, and 'journey\_score' varies daily per route. The 'journey\_score' appears standardised, i.e. popularity is relative to other routes. However, the variables are point in time, i.e. constant over the year, mainly focusing on socio-economic aspects of regions.

To me, this would imply that rather than looking at individual daily trips, which may be influenced by variables such as weather, sickness outbreaks across regions, fuel prices, etc., we are looking to identify more structural differences between regions.

Therefore, the research question I intend to answer is:

“Which structural socio-economic factors between regions most influence the relative popularity of commuting routes?”

This research question is not the only way to interpret this task, and it will, in turn, directly influence the decisions I intend to make for inferential design. For example, aggregating daily data over the year per route may be more appropriate than using the current daily set, as structural socio-economic factors such as average wages do not vary materially over the course of a week or month.

The submission is structured to focus on the key aspects of statistical inference and analytics. Code is included where helpful to explain the process, and excluded where not, however complete code can be found in the ‘Article Notebook’. Some elements of the process, such as explanatory data analysis, are found in the appendices and referenced throughout the submission. Each section summarises the key points, and a conclusion is provided at the end of the submission.

# 2. Inferential Design

|  |
| --- |
| Section Summary |
| * A cleaned version of the dataset is loaded in * **Data Refinement:**   + The dataset is refined for the research question by dropping:     - Routes with the same origin and destination     - Days of the year which are non-typical for commuting (i.e. unrepresentative data in terms of the research question)       * Weekends       * Bank Holidays * **Feature Additions:**   + Geospatial and rent cost features are added to the dataset to reduce omitted-variable bias (OVB) and improve model performance * **Data Aggregation:**   + Absolute differences for each variable are calculated between the two local authorities for each route   + The dataset is aggregated to one row per route, per local authority pair, per year * **Feature Transformations:**   + Skewness is identified in the data   + Discussion around addressing this depends on the model used |

|  |
| --- |
| Warning |
| This code follows a setup ([Appendix A - Setup: Section 6.1](#sec-setup)) and preliminary exploratory analysis of the data ([Appendix B - EDA: Section 6.2](#sec-eda)) in the Appendices.  The code in this section is intended to be run independently following that setup with the cleaned dataset, which can be downloaded from the [Github project repo](https://github.com/2601547/commute). However, the library installations must be first run and can be found in the [Appendix A - Packages: Section 6.1.2](#sec-packages) section.  To run independently from the original dataset, it is important to follow the step-by-step process in the [Appendices: Section 6](#sec-appendices) section, that precedes this section in terms of the running order. |

The cleaned (achieved in the [Appendices: Section 6](#sec-appendices)) dataset is loaded in:

base\_data\_dir = os.path.abspath(  
 '../../1. Data/LAcommute')  
commute = pd.read\_csv(  
 os.path.join(base\_data\_dir, "LAcommute\_clean.csv"), index\_col=0)  
  
commute

|  | date | area\_name\_origin | area\_code\_origin | area\_name\_dest | area\_code\_dest | journey\_score | journey\_count\_decile | distance | population\_origin | population\_dest | ... | gcse\_rate\_origin | life\_satisfaction\_origin | housing\_growth\_origin | value\_added\_hourly\_dest | median\_weekly\_pay\_dest | emp\_rate\_dest | travel\_time\_dest | gcse\_rate\_dest | life\_satisfaction\_dest | housing\_growth\_dest |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 2019-01-01 | Hartlepool | E06000001 | Hartlepool | E06000001 | 1.4414 | 9 | 0.000000 | 92401 | 92401 | ... | 67.6 | 7.33 | 161 | 28.31 | 487.4 | 67.1 | 12.9 | 67.6 | 7.33 | 161 |
| 1 | 2019-01-01 | Hartlepool | E06000001 | County Durham | E06000047 | -0.3129 | 3 | 37592.170378 | 92401 | 518562 | ... | 67.6 | 7.33 | 161 | 28.96 | 469.4 | 71.4 | 14.1 | 67.6 | 7.43 | 1343 |
| 2 | 2019-01-01 | Middlesbrough | E06000002 | Middlesbrough | E06000002 | 1.0253 | 10 | 0.000000 | 142134 | 142134 | ... | 63.2 | 7.21 | 456 | 29.30 | 420.8 | 65.6 | 15.4 | 63.2 | 7.21 | 456 |
| 3 | 2019-01-01 | Middlesbrough | E06000002 | Redcar and Cleveland | E06000003 | 0.3086 | 7 | 13069.176565 | 142134 | 136699 | ... | 63.2 | 7.21 | 456 | 26.54 | 439.2 | 68.4 | 13.3 | 69.6 | 7.44 | 365 |
| 4 | 2019-01-01 | Middlesbrough | E06000002 | Stockton-on-Tees | E06000004 | 0.3772 | 8 | 7379.212731 | 142134 | 196860 | ... | 63.2 | 7.21 | 456 | 34.37 | 469.4 | 74.8 | 13.2 | 69.5 | 7.40 | 616 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 301022 | 2019-12-31 | Westminster | E09000033 | Sutton | E09000029 | -0.6732 | 4 | 16964.439602 | 208415 | 208516 | ... | 77.3 | 7.21 | 580 | 35.19 | 565.8 | 77.4 | 9.0 | 82.0 | 7.36 | 313 |
| 301023 | 2019-12-31 | Westminster | E09000033 | Tower Hamlets | E09000030 | 1.3720 | 10 | 8616.142460 | 208415 | 305066 | ... | 77.3 | 7.21 | 580 | 60.46 | 680.3 | 74.4 | 4.4 | 72.4 | 7.13 | 3248 |
| 301024 | 2019-12-31 | Westminster | E09000033 | Waltham Forest | E09000031 | -0.4483 | 6 | 13672.865893 | 208415 | 281015 | ... | 77.3 | 7.21 | 580 | 34.63 | 624.7 | 71.5 | 7.2 | 71.5 | 7.30 | 1263 |
| 301025 | 2019-12-31 | Westminster | E09000033 | Wandsworth | E09000032 | 0.1871 | 8 | 7117.584240 | 208415 | 334558 | ... | 77.3 | 7.21 | 580 | 35.15 | 746.7 | 84.9 | 6.2 | 74.2 | 7.34 | 1415 |
| 301026 | 2019-12-31 | Westminster | E09000033 | Westminster | E09000033 | 1.6534 | 10 | 0.000000 | 208415 | 208415 | ... | 77.3 | 7.21 | 580 | 52.46 | 771.6 | 67.2 | 5.1 | 77.3 | 7.21 | 580 |

The cleaned dataset removes all routes with missing values e.g. “City of London” and “Isles of Scilly”. The full list of columns / variables can be seen below:

Index(['date', 'area\_name\_origin', 'area\_code\_origin', 'area\_name\_dest',  
 'area\_code\_dest', 'journey\_score', 'journey\_count\_decile', 'distance',  
 'population\_origin', 'population\_dest', 'value\_added\_hourly\_origin',  
 'median\_weekly\_pay\_origin', 'emp\_rate\_origin', 'travel\_time\_origin',  
 'gcse\_rate\_origin', 'life\_satisfaction\_origin', 'housing\_growth\_origin',  
 'value\_added\_hourly\_dest', 'median\_weekly\_pay\_dest', 'emp\_rate\_dest',  
 'travel\_time\_dest', 'gcse\_rate\_dest', 'life\_satisfaction\_dest',  
 'housing\_growth\_dest'],  
 dtype='object')

## 2.1 Data Refinement

|  |
| --- |
| Subsection Summary |
| * The dataset is refined for the reserach quesion by dropping:   + Routes with the same origin and destination   + Days of the year which are non-typical for commuting (i.e. unrepresentative data in terms of the research question)     - Weekends     - Bank Holidays |

My research question focuses on the structural differences between regions, and therefore, I will be using one 'journey\_score' per route, per local authority pair. This can also be achieved in multiple ways (e.g. taking data for each route for one day only, averaging over some time, etc.). I will use the average over the year, as I believe this is the most representative of the structural differences between regions and allows full use of the dataset. This also avoids any potential bias from seasonal effects (such as holidays), weather, or other temporal factors and instead smoothes out the data.

To aggregate this way, I will need to remove any data that is not representative of structural commuting route popularity. This includes removing days that are not typical for commuting, such as weekends and bank holidays.

* Weekends are not typical for commuting as most people do not work and therefore commute on weekend
* Bank holidays are not typical for commuting as most people do not work on bank holidays

One of the advantages of using the average over the year instead of a smaller subset is that these effects are minimised. However, it is still important to consider this potential source of bias in the data and remove it, as I want the model to be as representative as possible to get stronger confidence in the inference.

I drop all rows where there are weekends:

commute['date'] = commute['date'].apply(  
 pd.to\_datetime, format='%Y-%m-%d', errors='coerce')  
commute = commute[commute['date'].dt.dayofweek < 5]

Pull bank holidays from the [UK Government’s Bank Holiday API](https://www.gov.uk/bank-holidays.json), and drops all rows where the date is a bank holiday:

url = 'https://www.gov.uk/bank-holidays.json'  
response = requests.get(url)  
gov\_uk\_bank\_holdadays = response.json()  
  
eng\_wales\_holidays = gov\_uk\_bank\_holdadays['england-and-wales']['events'] # filter to england and wales...  
bank\_holidays = [pd.to\_datetime(event['date']) for event in eng\_wales\_holidays if '2019' in event['date']] # ... in 2019  
  
commute = commute[~commute['date'].isin(bank\_holidays)]

Similarly, there are routes in the dataset where the origin and destination are the same. These can easily be identified when 'distance' = 0 and capture employees commuting within the same local authority.

Since the research question is focused on regional socio-economic structural differences, these are irrelevant as there are no differences for intra-regional commuting. These routes are, therefore, dropped:

commute = commute[commute['distance'] != 0]

## 2.2 Feature Additions

|  |
| --- |
| Subsection Summary |
| * Geospatial, centrality and rent cost features are added to the dataset to reduce omitted-variable bias (OVB) and improve model performance |

The dataset provided is largely complete, however, omitted-variable bias (OVB) is a key concern in any inferential task and so additional features I believe may be useful are added. These are:

* Coordinates of the local authorities
* A centrality measure
* Accommodation rental costs

Once multicollinearity is assessed, these may need to be removed.

### 2.2.1 Geometry

LAs are regional by definition. Therefore, geospatial relationships may need to be considered, and coordinates are added to the dataset accordingly.

The coordinates for the LA can be found in the [government depo](https://geoportal.statistics.gov.uk/datasets/9cb3c710143649499ff6acaca927d205_0/explore), and is downloaded as the file "LAD\_Dec\_2019\_Boundaries\_UK\_BFC\_2022\_-5126023737554987305.csv".

LA\_coordinates = pd.read\_csv(  
 os.path.join(base\_data\_dir, "LAD\_Dec\_2019\_Boundaries\_UK\_BFC\_2022\_-5126023737554987305.csv"), index\_col=0)  
  
LA\_coordinates.head(3)

|  | objectid | lad19cd | lad19nm | lad19nmw | bng\_e | bng\_n | long | lat | st\_areasha | st\_lengths | Shape\_\_Area | Shape\_\_Length | GlobalID |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| FID |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 1 | 1 | E06000001 | Hartlepool |  | 447160 | 531474 | -1.27018 | 54.67614 | 9.371262e+07 | 71011.933949 | 2.797890e+08 | 122680.152623 | d3127426-63c8-4358-a554-b3fe4f0e537e |
| 2 | 2 | E06000002 | Middlesbrough |  | 451141 | 516887 | -1.21099 | 54.54467 | 5.388156e+07 | 44481.691242 | 1.598722e+08 | 76614.777246 | 7caac8a1-a10b-4878-a2ab-4fdb80e987a4 |
| 3 | 3 | E06000003 | Redcar and Cleveland |  | 464361 | 519597 | -1.00608 | 54.56752 | 2.450695e+08 | 96703.989701 | 7.274510e+08 | 166599.724875 | 96cc264b-6155-48cb-8fc0-7e41684d2682 |

def merge\_coordinates(df, coordinates, merge\_col, prefix):  
 df = df.merge(  
 coordinates[['lad19nm', 'lat', 'long']],  
 how='left',  
 left\_on=merge\_col,  
 right\_on='lad19nm'  
 )  
 df = df.rename(columns={  
 'lat': f'lat\_{prefix}',  
 'long': f'long\_{prefix}'  
 })  
 return df.drop(columns=['lad19nm'])

commute = merge\_coordinates(commute, LA\_coordinates, 'area\_name\_origin', 'origin')  
commute = merge\_coordinates(commute, LA\_coordinates, 'area\_name\_dest', 'dest')

### 2.2.2 Centrality

Centrality measures how connected a node is in a network. It may add insight into major hubs, act as a proxy for urban/rural splits, and soak up concerns around spatial autocorrelation.

This is literature-driven, as Tranos et al. (2015) demonstated that centrality play a key role in understanding international migration flows, and it is reasonable to believe this may also apply at a smaller scale.

I use an undirected graph (visualised in [Figure 5](#fig-centr)) and calulate a normalised centrality metric based on node connections (i.e. the number of connections divided by the maximum number of connections), as this can be done quickly with the data provided.

G = nx.Graph()  
for \_, row in commute\_centr.iterrows():  
 G.add\_edge(row['area\_name\_origin'], row['area\_name\_dest'])

degree\_centrality = nx.degree\_centrality(G)  
  
centrality = pd.DataFrame({  
 'area\_name': list(degree\_centrality.keys()),  
 'centrality': list(degree\_centrality.values())  
})

def merge\_centrality(df, centrality, merge\_col, prefix):  
 df = df.merge(  
 centrality,  
 how='left',  
 left\_on=merge\_col,  
 right\_on='area\_name'  
 )  
 df = df.rename(columns={  
 'centrality': f'centrality\_{prefix}'  
 })  
 return df.drop(columns=['area\_name'])

commute = merge\_centrality(commute, centrality, 'area\_name\_origin', 'origin')  
commute = merge\_centrality(commute, centrality, 'area\_name\_dest', 'dest')

|  | date | area\_name\_origin | area\_code\_origin | area\_name\_dest | area\_code\_dest | journey\_score | journey\_count\_decile | distance | population\_origin | population\_dest | ... | travel\_time\_dest | gcse\_rate\_dest | life\_satisfaction\_dest | housing\_growth\_dest | lat\_origin | long\_origin | lat\_dest | long\_dest | centrality\_origin | centrality\_dest |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 2019-01-02 | Hartlepool | E06000001 | Stockton-on-Tees | E06000004 | 1.5423 | 5 | 12951.008041 | 92401 | 196860 | ... | 13.2 | 69.5 | 7.40 | 616 | 54.67614 | -1.27018 | 54.556911 | -1.30664 | 0.025862 | 0.051724 |
| 1 | 2019-01-02 | Hartlepool | E06000001 | County Durham | E06000047 | 1.5683 | 5 | 37592.170378 | 92401 | 518562 | ... | 14.1 | 67.6 | 7.43 | 1343 | 54.67614 | -1.27018 | 54.685131 | -1.84050 | 0.025862 | 0.077586 |
| 2 | 2019-01-02 | Middlesbrough | E06000002 | Redcar and Cleveland | E06000003 | 0.9239 | 8 | 13069.176565 | 142134 | 136699 | ... | 13.3 | 69.6 | 7.44 | 365 | 54.54467 | -1.21099 | 54.567520 | -1.00608 | 0.043103 | 0.025862 |

### 2.2.3 Accommodation Rental Cost

Multiple studies (So et al., 2001; Ahrens and Lyons, 2021; Wander and Blumenberg, 2024) have shown that the cost of accommodation is a key factor in commuting decisions, using varying metrics such as house prices, rental fees, etc. I include rental costs as the data is available at the LA level in 2019 from the same data source as other variables (ONS). The metric can be used as a proxy for accommodation costs generally (i.e. house prices, which are likely to be correlated).

Data can be found here: [ONS Private Rental Market Summary](https://www.ons.gov.uk/peoplepopulationandcommunity/housing/bulletins/privaterentalmarketsummarystatisticsinengland/october2018toseptember2019).

rent = pd.read\_csv(  
 os.path.join(base\_data\_dir, "datadownload - extract.csv"))  
  
rent.head(3)

|  | LA Code1 | Area Code1 | Area | Room | Studio | One Bedroom | Two Bedrooms | Three Bedrooms | Four or more Bedrooms | All categories |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | NaN | E92000001 | ENGLAND | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| 1 | NaN | E12000001 | NORTH EAST | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| 2 | 1355.0 | E06000047 | County Durham | 368 | .. | 350 | 425 | 495 | 695 | 450 |

Columns are converted to the correct format. I select the "All categories" column (average rental cost for all properties), as otherwise, I would need to take a view on what kind of property to use, which may introduce bias.

def merge\_rent(df, rent, merge\_col, prefix):  
 df = df.merge(  
 rent[['Area', 'All categories']],  
 how='left',  
 left\_on=merge\_col,  
 right\_on='Area'  
 )  
 df = df.rename(columns={  
 'All categories': f'avg\_monthly\_rent\_{prefix}'  
 })  
 return df.drop(columns=['Area'])

commute = merge\_rent(commute, rent, 'area\_name\_origin', 'origin')  
commute = merge\_rent(commute, rent, 'area\_name\_dest', 'dest')

## 2.3 Data Aggregation

|  |
| --- |
| Subsection Summary |
| * Absolute differences for each variable are calculated between the two local authorities for each route * The dataset is aggregated to one row per route, per local authority pair, per year |

As per the research question, I take the average of the data over the year. This is done by grouping by the origin and destination local authorities, and taking the mean of the varying variables such as 'journey\_score' and 'avg\_monthly\_rent\_origin'. For the point in time variables, I take the first value as they are constant over the year.

|  | area\_name\_origin | area\_name\_dest | journey\_score | journey\_count\_decile | distance | population\_origin | population\_dest | value\_added\_hourly\_origin | median\_weekly\_pay\_origin | emp\_rate\_origin | ... | life\_satisfaction\_dest | housing\_growth\_dest | lat\_origin | long\_origin | lat\_dest | long\_dest | avg\_monthly\_rent\_origin | avg\_monthly\_rent\_dest | centrality\_origin | centrality\_dest |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | Barking and Dagenham | Brent | -0.707100 | 1.000000 | 27881.522854 | 218828 | 347424 | 36.89 | 523.5 | 67.3 | ... | 7.25 | 2404 | 51.545551 | 0.129479 | 51.564411 | -0.275680 | 1200.0 | 1452.0 | 0.189655 | 0.258621 |
| 1 | Barking and Dagenham | Camden | 0.040885 | 2.295455 | 20345.000863 | 218828 | 217136 | 36.89 | 523.5 | 67.3 | ... | 6.78 | 509 | 51.545551 | 0.129479 | 51.543060 | -0.162890 | 1200.0 | 2058.0 | 0.189655 | 0.336207 |
| 2 | Barking and Dagenham | Enfield | 0.020474 | 2.558559 | 19482.245346 | 218828 | 335151 | 36.89 | 523.5 | 67.3 | ... | 6.86 | 797 | 51.545551 | 0.129479 | 51.648880 | -0.081470 | 1200.0 | 1250.0 | 0.189655 | 0.189655 |
| 3 | Barking and Dagenham | Greenwich | 0.076071 | 2.899083 | 9642.046411 | 218828 | 288205 | 36.89 | 523.5 | 67.3 | ... | 7.22 | 1042 | 51.545551 | 0.129479 | 51.463928 | 0.050107 | 1200.0 | 1350.0 | 0.189655 | 0.241379 |
| 4 | Barking and Dagenham | Hackney | 0.017374 | 3.161074 | 13748.465842 | 218828 | 265825 | 36.89 | 523.5 | 67.3 | ... | 6.94 | 969 | 51.545551 | 0.129479 | 51.554920 | -0.060450 | 1200.0 | 1699.0 | 0.189655 | 0.224138 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 1257 | Wolverhampton | Telford and Wrekin | 0.120656 | 2.984615 | 29347.918605 | 263519 | 182081 | 27.86 | 481.3 | 72.5 | ... | 7.39 | 836 | 52.597881 | -2.127460 | 52.714169 | -2.489410 | 555.0 | 575.0 | 0.068966 | 0.043103 |
| 1258 | Wolverhampton | Walsall | 0.999048 | 8.206349 | 10596.257678 | 263519 | 284600 | 27.86 | 481.3 | 72.5 | ... | 7.35 | 145 | 52.597881 | -2.127460 | 52.605030 | -1.970440 | 555.0 | 550.0 | 0.068966 | 0.051724 |
| 1259 | York | East Riding of Yorkshire | 0.321147 | 4.313492 | 38473.087474 | 203877 | 338944 | 34.64 | 478.2 | 78.2 | ... | 7.74 | 1447 | 53.965820 | -1.073750 | 53.881119 | -0.661950 | 775.0 | 495.0 | 0.025862 | 0.051724 |
| 1260 | York | Leeds | 0.247854 | 4.144628 | 32762.280089 | 203877 | 804640 | 34.64 | 478.2 | 78.2 | ... | 7.28 | 2950 | 53.965820 | -1.073750 | 53.822731 | -1.507360 | 775.0 | 675.0 | 0.025862 | 0.103448 |
| 1261 | York | Wakefield | 0.000000 | 1.666667 | 40490.805449 | 203877 | 348201 | 34.64 | 478.2 | 78.2 | ... | 7.28 | 1227 | 53.965820 | -1.073750 | 53.659222 | -1.420920 | 775.0 | 525.0 | 0.025862 | 0.086207 |

### 2.3.1 Addressing Paired Data

The dataset is structured as paired routes, i.e. for most A to B routes, there is also a B to A route on the same day. This makes sense as a) employees may commute in both directions and b) commuting data may include the return journey.

However, it presents a problem for this specific dataset, as the 'journey\_score' is not symmetric/directional but is instead a raw measure of popularity, whereas almost all the other variables are in the context of origin/dest directions. The only exception is 'distance', which is the same in both directions.   Without addressing this dataset structure, the model will be confused by positive and similar 'journey\_scores' where all other variables are equal and opposite.

For example, for the route between “Bristol” and “Bath”:

bristol\_to\_bath = commute[  
 ((commute['area\_name\_origin'] == 'Bristol, City of') & (commute['area\_name\_dest'] == 'Bath and North East Somerset')) |  
 ((commute['area\_name\_origin'] == 'Bath and North East Somerset') & (commute['area\_name\_dest'] == 'Bristol, City of'))  
]  
  
bristol\_to\_bath

|  | area\_name\_origin | area\_name\_dest | journey\_score | journey\_count\_decile | distance | population\_origin | population\_dest | value\_added\_hourly\_origin | median\_weekly\_pay\_origin | emp\_rate\_origin | ... | life\_satisfaction\_dest | housing\_growth\_dest | lat\_origin | long\_origin | lat\_dest | long\_dest | avg\_monthly\_rent\_origin | avg\_monthly\_rent\_dest | centrality\_origin | centrality\_dest |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 48 | Bath and North East Somerset | Bristol, City of | 0.811427 | 6.350598 | 15196.650331 | 190176 | 469920 | 28.73 | 459.8 | 82.0 | ... | 7.15 | 1621 | 51.356041 | -2.48654 | 51.471149 | -2.57742 | 990.0 | 975.0 | 0.034483 | 0.051724 |
| 132 | Bristol, City of | Bath and North East Somerset | 0.844441 | 6.519841 | 15196.650331 | 469920 | 190176 | 31.01 | 528.3 | 77.1 | ... | 7.29 | 764 | 51.471149 | -2.57742 | 51.356041 | -2.48654 | 975.0 | 990.0 | 0.051724 | 0.034483 |

Here, we can see that the 'journey\_score' is approximately the same in both directions, but all other variables (excluding 'distance') are flipped.

This would directly influence the model interpretation, since 'distance' is the only symmetric variable, it would be the only variable that could feasibly explain variation in the 'journey\_score' and its importance would be inflated.

### 2.3.2 Calculating Differences

Given the research question, I believe the differences between the two regions are more important than the absolute variables in explaining the popularity of the routes. This is because they reflect a gradient/contrast effect between the regions, likely influencing patterns more directly than isolated conditions within either region.

In plain terms, a person within a region with a high average salary is perhaps less likely to commute to another region with a high average wage relative to a person within a region with a low average wage, i.e. the relative difference between the two regions is more important. Again, this is a subjective, researcher-driven decision as opposed to the only way to do it.

To obtain directionless variables, I take the absolute difference between the origin and destination (and then drop all the original columns). This ensures they are the same for both route directions, and therefore, any model is not confused by the direction of the route.

columns\_to\_diff = [  
 'population',  
 'value\_added\_hourly',  
 'median\_weekly\_pay',  
 'emp\_rate',  
 'travel\_time',  
 'gcse\_rate',  
 'life\_satisfaction',  
 'housing\_growth',  
 'avg\_monthly\_rent',  
 'centrality'  
]  
  
for col in columns\_to\_diff:  
 commute[f'|\_{col}\_diff\_|'] = abs( # | x | to display the absolute value of the difference  
 commute[f'{col}\_origin'] - commute[f'{col}\_dest'])

For geometric variables, I will the midpoint of the origin and destination local authorities, as this can later be used so consider spatial relationships between routes.

commute['midpoint\_long'] = (commute['long\_origin'] + commute['long\_dest']) / 2  
commute['midpoint\_lat'] = (commute['lat\_origin'] + commute['lat\_dest']) / 2  
  
commute['route\_midpoint\_(geo)'] = gpd.points\_from\_xy(commute['midpoint\_long'], commute['midpoint\_lat'])  
  
commute = commute.drop(columns=['long\_origin', 'lat\_origin', 'long\_dest', 'lat\_dest', 'midpoint\_long', 'midpoint\_lat',])  
  
commute.head(3)

|  | area\_name\_origin | area\_name\_dest | journey\_score | journey\_count\_decile | distance | |\_population\_diff\_| | |\_value\_added\_hourly\_diff\_| | |\_median\_weekly\_pay\_diff\_| | |\_emp\_rate\_diff\_| | |\_travel\_time\_diff\_| | |\_gcse\_rate\_diff\_| | |\_life\_satisfaction\_diff\_| | |\_housing\_growth\_diff\_| | |\_avg\_monthly\_rent\_diff\_| | |\_centrality\_diff\_| | route\_midpoint\_(geo) |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | Barking and Dagenham | Brent | -0.707100 | 1.000000 | 27881.522854 | 128596 | 0.07 | 29.6 | 3.1 | 0.9 | 3.5 | 0.10 | 1356 | 252.0 | 0.068966 | POINT (-0.07310 51.55498) |
| 1 | Barking and Dagenham | Camden | 0.040885 | 2.295455 | 20345.000863 | 1692 | 14.43 | 170.7 | 2.3 | 2.7 | 4.3 | 0.57 | 539 | 858.0 | 0.146552 | POINT (-0.01671 51.54431) |
| 2 | Barking and Dagenham | Enfield | 0.020474 | 2.558559 | 19482.245346 | 116323 | 5.36 | 15.6 | 2.5 | 0.1 | 0.7 | 0.49 | 251 | 50.0 | 0.000000 | POINT (0.02400 51.59722) |

### 2.3.3 Removing Pairs

I remove the paired routes, replacing them with a single route with the average of the two routes. Given the similarity of the “journey scores” between routes, this is a reasonable assumption to make, with the caveat that where there are large differences, there is a loss of information and potential bias. However, exploration of the data shows that this is rarely an issue, as the data seems to include journeys to and from the same local authority.

The dataset following this process looks like:

|  | pairs | journey\_score | journey\_count\_decile | distance | |\_population\_diff\_| | |\_value\_added\_hourly\_diff\_| | |\_median\_weekly\_pay\_diff\_| | |\_emp\_rate\_diff\_| | |\_travel\_time\_diff\_| | |\_gcse\_rate\_diff\_| | |\_life\_satisfaction\_diff\_| | |\_housing\_growth\_diff\_| | |\_avg\_monthly\_rent\_diff\_| | |\_centrality\_diff\_| | route\_midpoint\_(geo) |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | Barking and Dagenham - Barnet | -0.010609 | 1.363636 | 25056.274464 | 171762 | 0.09 | 51.4 | 8.3 | 0.9 | 13.4 | 0.05 | 1202 | 150.0 | 0.017241 | POINT (-0.04437 51.57832) |
| 1 | Barking and Dagenham - Brent | -0.397323 | 1.291667 | 27881.522854 | 128596 | 0.07 | 29.6 | 3.1 | 0.9 | 3.5 | 0.10 | 1356 | 252.0 | 0.068966 | POINT (-0.07310 51.55498) |
| 2 | Barking and Dagenham - Camden | 0.112844 | 2.959443 | 20345.000863 | 1692 | 14.43 | 170.7 | 2.3 | 2.7 | 4.3 | 0.57 | 539 | 858.0 | 0.146552 | POINT (-0.01671 51.54431) |
| 3 | Barking and Dagenham - Enfield | -0.004368 | 2.500977 | 19482.245346 | 116323 | 5.36 | 15.6 | 2.5 | 0.1 | 0.7 | 0.49 | 251 | 50.0 | 0.000000 | POINT (0.02400 51.59722) |
| 4 | Barking and Dagenham - Greenwich | 0.114385 | 2.973163 | 9642.046411 | 69377 | 1.56 | 95.4 | 8.3 | 0.0 | 0.8 | 0.13 | 6 | 150.0 | 0.051724 | POINT (0.08979 51.50474) |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 672 | West Berkshire - Wokingham | 0.163863 | 2.844894 | 29259.923429 | 10906 | 4.18 | 64.6 | 5.8 | 1.6 | 5.6 | 0.01 | 512 | 175.0 | 0.008621 | POINT (-1.08649 51.43427) |
| 673 | Westminster - Wiltshire | 0.075996 | 1.931034 | 126401.984988 | 294570 | 21.88 | 290.4 | 10.8 | 9.5 | 4.9 | 0.29 | 2266 | 1633.0 | 0.344828 | POINT (-1.03978 51.42052) |
| 674 | Westminster - Windsor and Maidenhead | 0.026557 | 2.449654 | 37615.785116 | 54522 | 0.44 | 164.8 | 11.9 | 4.6 | 0.8 | 0.29 | 270 | 1183.0 | 0.318966 | POINT (-0.41418 51.49627) |
| 675 | Westminster - Wokingham | -0.168467 | 1.416667 | 51144.513300 | 36626 | 4.76 | 113.0 | 10.0 | 6.8 | 3.1 | 0.12 | 549 | 1283.0 | 0.344828 | POINT (-0.52615 51.46758) |
| 676 | Windsor and Maidenhead - Wokingham | 0.161748 | 2.881737 | 14209.316238 | 17896 | 5.20 | 51.8 | 1.9 | 2.2 | 2.3 | 0.17 | 819 | 100.0 | 0.025862 | POINT (-0.78738 51.45165) |

i.e. 677 unqiue directionless routes.

## 2.4 Feature Transformations

|  |
| --- |
| Subsection Summary |
| * Skenewness is identified in the data * Discussion around addressing this depend on the model used |

Preliminary exploratory data analysis highlighted the following key points:

* The 'journey\_score' variable is not normally distributed
* No explanatory variables are normally distributed
  + Generally are clustered around 0

Both these poitns are visualised in [Figure 6](#fig-histogram), full details can be found in [Appendix B - EDA 6.2](#sec-eda).

I will proceed with analytical decisions with this in mind going forward on a model by model basis, considering the use of transformations where appropriate.

# 3. Linear Regression (OLS and SEM)

My primary method to address the research question will be linear regressions, as they generally have the most inferential power and are simple to interpret/understand. OLS coefficients can provide data on both the strength of a relationship (what our research question is asking), the direction of the relationship (positive or negative), and also the effect size (i.e. how much the dependent variable changes for a unit change in the independent variable) holding everything else constant.

Other models generally can provide information only on relative importance. However, this is a developing area with analytical innovations such as Shapley values, which can provide a consistent measure of relative importance across models (Buckmann & Joseph, 2022). This is important to note, but not considered here for simplicity.

The linear regression/OLS method relies on a number of important assumptions, which, if not met, can lead to bias and incorrect inference.

As is common practice, I will use a 95% confidence level to determine significance, but other levels may also be discussed.

|  |
| --- |
| Further Information on Linear Regression / OLS |
| The form of the model is:  Where:   * is the dependent variable (i.e. the variable we are trying to predict) * are the independent variables (i.e. the variables we are using to predict ) * is the intercept (i.e. the value of when all are 0) * are the coefficients (i.e. the effect of each on ) * is the error term (i.e. the difference between the predicted and actual value of )   Assumptions   * Linearity: The regression model is linear in parameters (coefficients) * Zero mean of errors: The expected value of the error term is zero * Homoscedasticity: The error term has constant variance across all levels of the independent variables * Independence of errors: The error terms are uncorrelated with each other i.e. no autocorrelation * No perfect multicollinearity: Independent variables are not perfectly linearly related i.e. no redundant predictors   Impact on inference if failed:   * Linearity: The the estimated effects will be biased and therefore unreliable * Zero mean of errors: The constant term absorbs systematic error, leading to biasing coefficient estimates * Homoscedasticity: Standard errors will be incorrect, making inference of p-values and confidence intervals potentially incorrect when determining statistical significance * Independence of errors: Standard errors are underestimated, which inflates statistical significance metrics again leading to incorrect conclusions * No endogeneity: Coefficient estimates are biased and therefore unreliable when determining the effect of independent variables on the dependent variable |

I create a new data frame, so any transformations specific to OLS are not carried forward. This will be the working standard going forward for each model:

commuteOLS = commute.copy()

Variables are then standardised to ensure they are all on the same scale. This is important for OLS, as the coefficients are directly comparable.

|  | pairs | journey\_score | journey\_count\_decile | distance | |\_population\_diff\_| | |\_value\_added\_hourly\_diff\_| | |\_median\_weekly\_pay\_diff\_| | |\_emp\_rate\_diff\_| | |\_travel\_time\_diff\_| | |\_gcse\_rate\_diff\_| | |\_life\_satisfaction\_diff\_| | |\_housing\_growth\_diff\_| | |\_avg\_monthly\_rent\_diff\_| | |\_centrality\_diff\_| | route\_midpoint\_(geo) |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | Barking and Dagenham - Barnet | -0.010609 | 1.363636 | 0.084152 | 0.405396 | -1.148665 | -0.281985 | 1.081910 | -0.791513 | 1.930812 | -0.951542 | 0.337307 | -0.490285 | -0.749848 | POINT (-0.04437 51.57832) |
| 1 | Barking and Dagenham - Brent | -0.397323 | 1.291667 | 0.206818 | 0.102728 | -1.151899 | -0.632628 | -0.419492 | -0.791513 | -0.525323 | -0.633139 | 0.522719 | -0.205973 | -0.053169 | POINT (-0.07310 51.55498) |
| 2 | Barking and Dagenham - Camden | 0.112844 | 2.959443 | -0.120401 | -0.787089 | 1.170277 | 1.636900 | -0.650477 | 0.243732 | -0.326847 | 2.359855 | -0.460928 | 1.483174 | 0.991851 | POINT (-0.01671 51.54431) |

## 3.1 Simple Linear Regression

The most straightforward approach is to use variables as they are and run a simple linear regression.

Geometry is not included as point coordinates are non-numeric and, therefore, cannot be directly used in OLS.

y = commuteOLS['journey\_score']  
X = commuteOLS[['distance', '|\_population\_diff\_|', '|\_value\_added\_hourly\_diff\_|',  
 '|\_median\_weekly\_pay\_diff\_|', '|\_emp\_rate\_diff\_|',  
 '|\_travel\_time\_diff\_|', '|\_gcse\_rate\_diff\_|',  
 '|\_life\_satisfaction\_diff\_|', '|\_housing\_growth\_diff\_|',  
 '|\_avg\_monthly\_rent\_diff\_|', '|\_centrality\_diff\_|']]  
X = sm.add\_constant(X)  
  
lr\_model = sm.OLS(y, X).fit()  
  
print(lr\_model.summary())

OLS Regression Results   
==============================================================================  
Dep. Variable: journey\_score R-squared: 0.201  
Model: OLS Adj. R-squared: 0.188  
Method: Least Squares F-statistic: 15.21  
Date: Mon, 05 May 2025 Prob (F-statistic): 1.33e-26  
Time: 14:31:04 Log-Likelihood: -284.09  
No. Observations: 677 AIC: 592.2  
Df Residuals: 665 BIC: 646.4  
Df Model: 11   
Covariance Type: nonrobust   
===============================================================================================  
 coef std err t P>|t| [0.025 0.975]  
-----------------------------------------------------------------------------------------------  
const 0.3031 0.014 21.231 0.000 0.275 0.331  
distance -0.1696 0.017 -10.152 0.000 -0.202 -0.137  
|\_population\_diff\_| 0.0573 0.018 3.239 0.001 0.023 0.092  
|\_value\_added\_hourly\_diff\_| 0.0175 0.016 1.093 0.275 -0.014 0.049  
|\_median\_weekly\_pay\_diff\_| -0.0238 0.022 -1.080 0.281 -0.067 0.019  
|\_emp\_rate\_diff\_| 0.0073 0.015 0.473 0.636 -0.023 0.038  
|\_travel\_time\_diff\_| -0.0482 0.018 -2.715 0.007 -0.083 -0.013  
|\_gcse\_rate\_diff\_| -0.0108 0.015 -0.745 0.457 -0.039 0.018  
|\_life\_satisfaction\_diff\_| -0.0058 0.015 -0.389 0.697 -0.035 0.023  
|\_housing\_growth\_diff\_| 0.0435 0.017 2.505 0.012 0.009 0.078  
|\_avg\_monthly\_rent\_diff\_| 0.0622 0.030 2.091 0.037 0.004 0.121  
|\_centrality\_diff\_| -0.0408 0.022 -1.823 0.069 -0.085 0.003  
==============================================================================  
Omnibus: 69.179 Durbin-Watson: 1.932  
Prob(Omnibus): 0.000 Jarque-Bera (JB): 88.132  
Skew: 0.864 Prob(JB): 7.29e-20  
Kurtosis: 3.376 Cond. No. 4.60  
==============================================================================  
  
Notes:  
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

### 3.1.1 Interpretation of the OLS results

General OLS considerations:

* R-squared = 0.201, indicates that c.20.1% of the variation in 'journey\_score' is explained by the linear combination of predictors
  + This isn’t particularly important for inference (relative to prediction) as long as the coefficients are statistically significant, but still a helpful metric
* F-statistic = 15.21 (p = c.0) tests the null hypothesis that all regression coefficients are jointly zero
  + Given the p-value is effectively zero, we reject the null hypothesis at all conventional confidence levels, i.e. the model is statistically significant for explaining some aspect of the variation

Key Variable:

* 'distance', is the most influential with the largest absolute coefficient (0.1696)
  + This implies for every unit standard deviation (std) increase in 'distance', the 'journey\_score' decreases by 0.1696 stds
    - Holding other variables constant
    - With a 95% confidence interval of [-0.202, -0.137]
* It is statistically significant (p < 0.001), and therefore, we can be confident that this is a real effect, i.e. not due to random chance
* The narrow confidence interval and large t-statistic imply a precisely estimated and statistically reliable effect

Other variables:

* From the OLS summary, the significant variables ranked (by absolute size of coefficient) are as follows:

| # | Variable | Coefficient (β) | 95% CI | p-value | Statistical Inference |
| --- | --- | --- | --- | --- | --- |
| 1 | distance | -0.1696 | [-0.202, -0.137] | 0.000 | Strong negative effect, highly significant and precisely estimated |
| 2 | avg\_monthly\_rent\_diff | +0.0622 | [0.004, 0.121] | 0.037 | Positive effect, significant (at 5%, but not 1 or 0.1%) and less precise |
| 3 | population\_diff | +0.0573 | [0.023, 0.092] | 0.001 | Positive effect, significant and generally precise |
| 4 | travel\_time\_diff | -0.0482 | [-0.083, -0.013] | 0.007 | Negative effect, significant and less precise |
| 5 | housing\_growth\_diff | +0.0435 | [0.009, 0.078] | 0.012 | Positive effect, significant (but not at 1 or 0.1%) and generally precise |
| - | All others | - | CI include 0 | >0.05 | No statistical significant 95% confidence |

Another metric that can be used is permutation importance, which is a measure of how much the model performance decreases when a variable is shuffled randomly and dropped. Therefore, it measures a distinct concept for which provides less inferential power than coefficients. However, it is helpful as a) it can be directly compared to other models and b) it is more robust to multicollinearity:

y\_pred = lr\_model.predict(X)  
baseline\_r2 = r2\_score(y, y\_pred)  
  
feature\_importances = {}  
  
for col in X.columns[1:]:  
 np.random.seed(0) # for reproducibility  
 X\_permuted = X.copy()  
 X\_permuted[col] = np.random.permutation(X\_permuted[col])  
 y\_permuted\_pred = lr\_model.predict(X\_permuted)  
 permuted\_r2 = r2\_score(y, y\_permuted\_pred)  
 feature\_importances[col] = baseline\_r2 - permuted\_r2  
  
perm\_importance = pd.DataFrame(list(feature\_importances.items()), columns=[  
 'Variable', 'Permutation Importance'])  
perm\_importance = perm\_importance.sort\_values(  
 by='Permutation Importance', ascending=False)  
  
perm\_importance.head() # only need to show top 5 to compare

|  | Variable | Permutation Importance |
| --- | --- | --- |
| 0 | distance | 0.308688 |
| 9 | |\_avg\_monthly\_rent\_diff\_| | 0.051189 |
| 1 | |\_population\_diff\_| | 0.039451 |
| 8 | |\_housing\_growth\_diff\_| | 0.024539 |
| 5 | |\_travel\_time\_diff\_| | 0.023728 |

The results corroborate that of the OLS for the five most influential variables and are generally consistent with the order, with the caveat that with random shuffling, the results are not as precise as OLS and can vary across runs.

### 3.1.2 Discussion

Most of the relationships are intuitively what we would expect. For example, a larger 'distance' is associated with a more popular commute route. This, in economic theory, is often attributed to the cost of commuting, as the benefits of commuting (e.g. higher wages) are outweighed by e.g. time, money, and effort (So et al., 2001). The statistic is precisely estimated, which adds confidence to the interpretation, however, preliminary exploratory data analysis in [Appendix B: Section 6.2](#sec-eda)) shows that the relationship between 'journey\_score' and 'distance' is non-linear (as seen in [Figure 8](#fig-scatterplot)), which violates the linearity assumption of OLS. This is a key consideration for accurate interpretation of the results, as the confidence interval-based precision assumes this assumption is met.

Similarly, larger 'avg\_monthly\_rent\_diff' differences between regions are associated with more popular the routes between them. This result corroborates the findings of Ahrens and Lyons (2021), who identified the same relationship between rent and commuting popularity in their study of Ireland. Since this was a researcher-driven feature addition, it suggests that OVB may have been reduced with its inclusion. This is in contrast to the 'centrality' metric added, which was not significant at any confidence level.

Other relationships are less intuitive, and their interpretation is nuanced. For example,  'travel\_time' represents the average time taken within a local authority to commute to work. The OLS regression results show that a larger difference in 'travel\_time' between regions is associated with a less popular route. This is counter-intuitive but may reflect that the regions with larger differences in travel time are also those with larger differences in other factors such as 'avg\_monthly\_rent\_diff', which may be driving the popularity of the route. If this is the case, there could be an issue of multicollinearity, which would violate the respective OLS assumption.

'housing\_growth\_diff' reflects a similar relationship 'avg\_monthly\_rent\_diff', and the explanatory factors may be similar. However, housing growth, unlike rent, is a lagging factor i.e. even within the context of annual data, house building can take a significant amount of time and is often driven by demand. This could confuse identifying the cause of the relationship - does housing growth influence commuting popularity, or does commuting popularity influence housing growth?

Final considerations:

Within the discussion, key elements of:

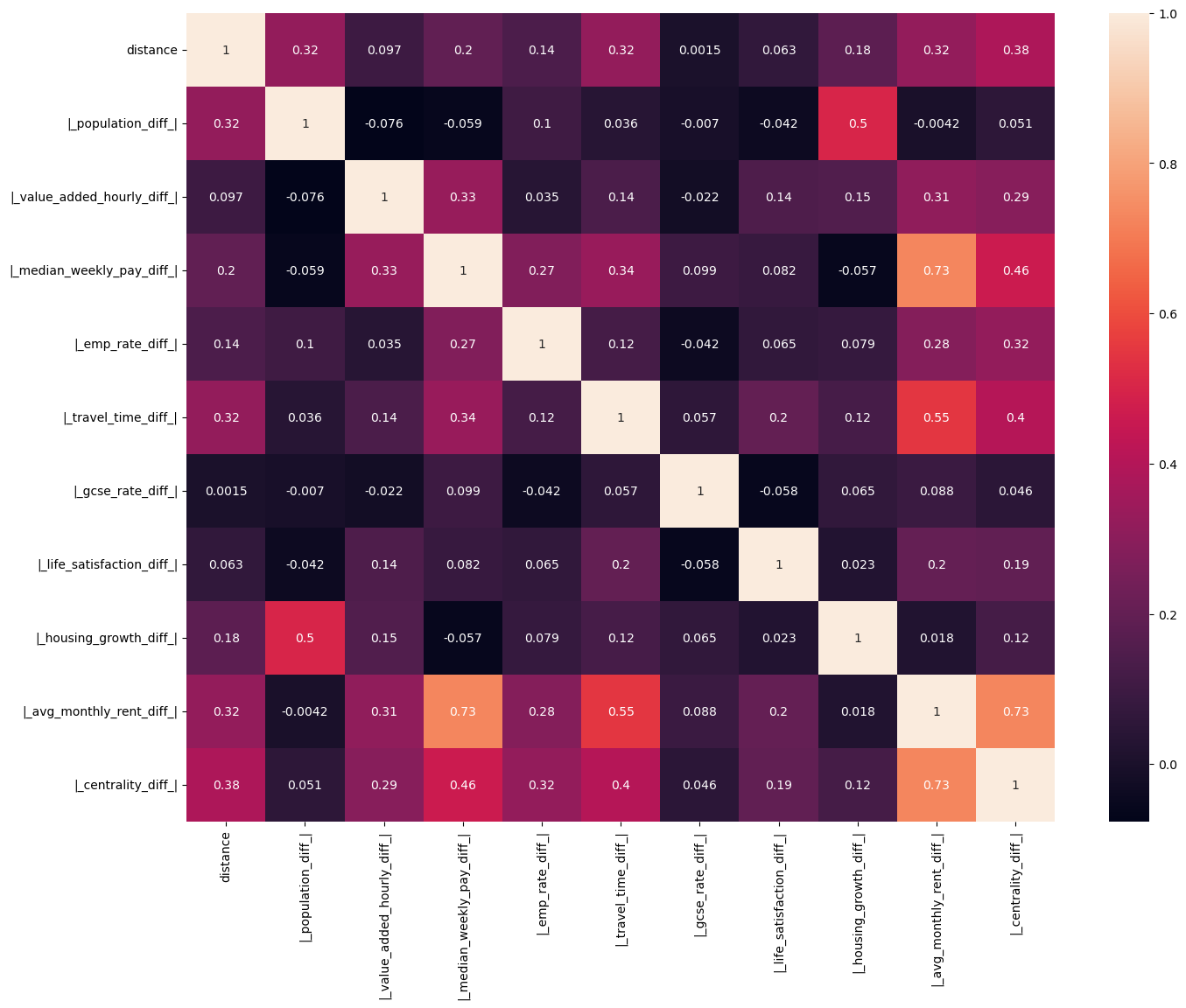
* Skewed/non-linear data
* Multicollinearity

which impact the ability to successfully draw an inference to address our research question. The simple OLS regression model is a good starting point, but it is not robust enough to draw a strong conclusion that the five listed variables are the most important factors.

### 3.1.3 Multicollinearity

We can test for collinearity between variables with a correlation matrix, and then use a variance inflation factor (VIF) to assess the impact of collinearity on the model:

commuteOLScm = commuteOLS.drop(columns=['pairs', 'journey\_score', 'journey\_count\_decile', 'route\_midpoint\_(geo)'])  
  
plt.figure(figsize=(16, 12))   
sns.heatmap(commuteOLScm.corr(), annot=True)



vif = pd.DataFrame()  
vif['Variable'] = X.columns  
vif['VIF'] = [variance\_inflation\_factor(X.values, i) for i in range(X.shape[1])]  
  
vif

|  | Variable | VIF |
| --- | --- | --- |
| 0 | const | 1.000000 |
| 1 | distance | 1.369576 |
| 2 | |\_population\_diff\_| | 1.534792 |
| 3 | |\_value\_added\_hourly\_diff\_| | 1.261580 |
| 4 | |\_median\_weekly\_pay\_diff\_| | 2.389630 |
| 5 | |\_emp\_rate\_diff\_| | 1.178689 |
| 6 | |\_travel\_time\_diff\_| | 1.546928 |
| 7 | |\_gcse\_rate\_diff\_| | 1.037265 |
| 8 | |\_life\_satisfaction\_diff\_| | 1.085963 |
| 9 | |\_housing\_growth\_diff\_| | 1.481976 |
| 10 | |\_avg\_monthly\_rent\_diff\_| | 4.340401 |
| 11 | |\_centrality\_diff\_| | 2.460915 |

The correlation matrix shows limited collinearity between the variables, with the greatest correlation between 'avg\_monthly\_rent\_diff' and 'median\_weekly\_pay\_diff' (0.73). This is not unexpected, as areas with higher rents will generally require a higher wage rate to afford the rent.

The VIF analysis corroborates this limited collinearity, as all values are below 10, a common concern threshold. All values are above 1, which indicates some level of collinearity present, again with the greatest values being 'avg\_monthly\_rent\_diff' (4.3) and 'median\_weekly\_pay\_diff' (2.4).

Multicollinearity violates the OLS assumptions and can lead to biased coefficient estimates, so it is important to accommodate for this. The OLS results show that 'avg\_monthly\_rent\_diff' is more significant than 'median\_weekly\_pay\_diff', has a larger coefficient, and is more precise, however, the VIF results show it also has greater collinearity with other variables. To optimise for the OLS assumptions, and to avoid the 'avg\_monthly\_rent\_diff' soaking up explanatory power from correlated variables, I will drop it from the model in future iterations. This is, again, a researcher-driven, subjective decision.

## 3.2 Refined Linear Regression

Multiple paths can be taken to address the data’s skewness (shown in [Figure 6](#fig-histogram)). The most common is to use a log transformation. However, the absolute difference variables include zero values, which do not work with a log transformation. Therefore, a Yeo-Johnson transformation is used, a generalisation of the log transformation that can handle negative values (Yeo and Johnson, 2000).

columns\_OLSr = [col for col in commuteOLSr.columns if col not in ['journey\_score', 'pairs', 'route\_midpoint\_(geo)']]  
transformer = PowerTransformer(method='yeo-johnson', standardize=True)  
commuteOLSr[columns\_OLSr] = transformer.fit\_transform(commuteOLSr[columns\_OLSr])

And running a new OLS regression with the transformed data (with dropped 'avg\_monthly\_rent\_diff'):

y = commuteOLSr['journey\_score']  
X = commuteOLSr[['distance', '|\_population\_diff\_|', '|\_value\_added\_hourly\_diff\_|',  
 '|\_median\_weekly\_pay\_diff\_|', '|\_emp\_rate\_diff\_|',  
 '|\_travel\_time\_diff\_|', '|\_gcse\_rate\_diff\_|',  
 '|\_life\_satisfaction\_diff\_|', '|\_housing\_growth\_diff\_|',  
 '|\_centrality\_diff\_|']]  
X = sm.add\_constant(X)  
  
rlr\_model = sm.OLS(y, X).fit()  
  
print(rlr\_model.summary())

OLS Regression Results   
==============================================================================  
Dep. Variable: journey\_score R-squared: 0.438  
Model: OLS Adj. R-squared: 0.430  
Method: Least Squares F-statistic: 51.99  
Date: Mon, 05 May 2025 Prob (F-statistic): 7.24e-77  
Time: 14:31:05 Log-Likelihood: -164.77  
No. Observations: 677 AIC: 351.5  
Df Residuals: 666 BIC: 401.2  
Df Model: 10   
Covariance Type: nonrobust   
===============================================================================================  
 coef std err t P>|t| [0.025 0.975]  
-----------------------------------------------------------------------------------------------  
const 0.3031 0.012 25.343 0.000 0.280 0.327  
distance -0.2759 0.013 -21.054 0.000 -0.302 -0.250  
|\_population\_diff\_| 0.0593 0.013 4.449 0.000 0.033 0.086  
|\_value\_added\_hourly\_diff\_| -0.0050 0.013 -0.387 0.699 -0.031 0.021  
|\_median\_weekly\_pay\_diff\_| -0.0367 0.014 -2.649 0.008 -0.064 -0.009  
|\_emp\_rate\_diff\_| 0.0188 0.013 1.494 0.136 -0.006 0.044  
|\_travel\_time\_diff\_| 0.0180 0.014 1.294 0.196 -0.009 0.045  
|\_gcse\_rate\_diff\_| -0.0139 0.012 -1.148 0.252 -0.038 0.010  
|\_life\_satisfaction\_diff\_| -0.0079 0.012 -0.635 0.525 -0.032 0.016  
|\_housing\_growth\_diff\_| 0.0373 0.013 2.806 0.005 0.011 0.063  
|\_centrality\_diff\_| -0.0192 0.014 -1.390 0.165 -0.046 0.008  
==============================================================================  
Omnibus: 60.620 Durbin-Watson: 1.604  
Prob(Omnibus): 0.000 Jarque-Bera (JB): 75.204  
Skew: 0.764 Prob(JB): 4.67e-17  
Kurtosis: 3.573 Cond. No. 2.14  
==============================================================================  
  
Notes:  
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

### 3.2.1 Interpretation of the OLS results

General OLS considerations:

* R-squared = 0.438, indicates that c.43.8% of the variation in 'journey\_score' is explained by the linear combination of predictors, an improvement from the previous model (c.20.1%)
  + This is helpful for inference, as it suggests that the model is better at explaining the variation in the data, and therefore the coefficients are more reliable
* F-statistic = 51.99 (p = c.0)
  + Given the p-value is effectively zero, the model is again statistically significant for explaining some aspects of the variation

Key Variable:

* 'distance', is the most influential with the largest coefficient (0.2759), an even larger effect than the previous model (+62.7%)
  + As a result of the transformation, we can no longer make an inference on the marginal impact of a unit change without back-transforming, which is a trade-off
  + It is statistically significant (p < 0.001), and therefore, again we can be confident that this is a real effect
  + The narrow confidence interval and large t-statistic implies a precisely estimated and statistically reliable effect

Other variables:

* With the new specification, 'travel\_time\_diff' is no longer significant at a 95% confidence level
* From the OLS summary, the significant variables ranked (by absolute size of coefficient) are as follows:

| # | Variable | Coefficient (β) | 95% CI | p-value | Relative to Simple OLS |
| --- | --- | --- | --- | --- | --- |
| 1 | distance | -0.2728 | [-0.302, -0.250] | 0.000 | Stronger negative effect, still highly significant and precisely estimated |
| 2 | population\_diff | +0.0593 | [0.033, 0.086] | 0.000 | Stronger positive effect, significant and generally precise |
| 3 | housing\_growth\_diff | +0.0373 | [0.011, 0.063] | 0.005 | Weaker positive effect, significant (but not at 0.1%) and generally precise |
| 4 | median\_weekly\_pay\_diff | -0.0367 | [-0.064, -0.009] | 0.001 | Stronger negative effect, now significant |
| - | All others | - | CI include 0 | >0.05 | No statistical significant 95% confidence |

The addition of 'median\_weekly\_pay\_diff' suggests that the correlation with 'avg\_monthly\_rent\_diff' may have impacted the inference of the previous model, whereas the exclusion of 'travel\_time\_diff' may reflect the change in distribution following the transformation.

Following this, permutation importance:

y\_pred = rlr\_model.predict(X)  
baseline\_r2 = r2\_score(y, y\_pred)  
  
feature\_importances = {}  
  
for col in X.columns[1:]:  
 np.random.seed(0) # for reproducibility  
 X\_permuted = X.copy()  
 X\_permuted[col] = np.random.permutation(X\_permuted[col])  
 y\_permuted\_pred = rlr\_model.predict(X\_permuted)  
 permuted\_r2 = r2\_score(y, y\_permuted\_pred)  
 feature\_importances[col] = baseline\_r2 - permuted\_r2  
  
perm\_importance = pd.DataFrame(list(feature\_importances.items()), columns=[  
 'Variable', 'Permutation Importance'])  
perm\_importance = perm\_importance.sort\_values(  
 by='Permutation Importance', ascending=False)  
  
perm\_importance.head(4) # only need to show top 4 to compare

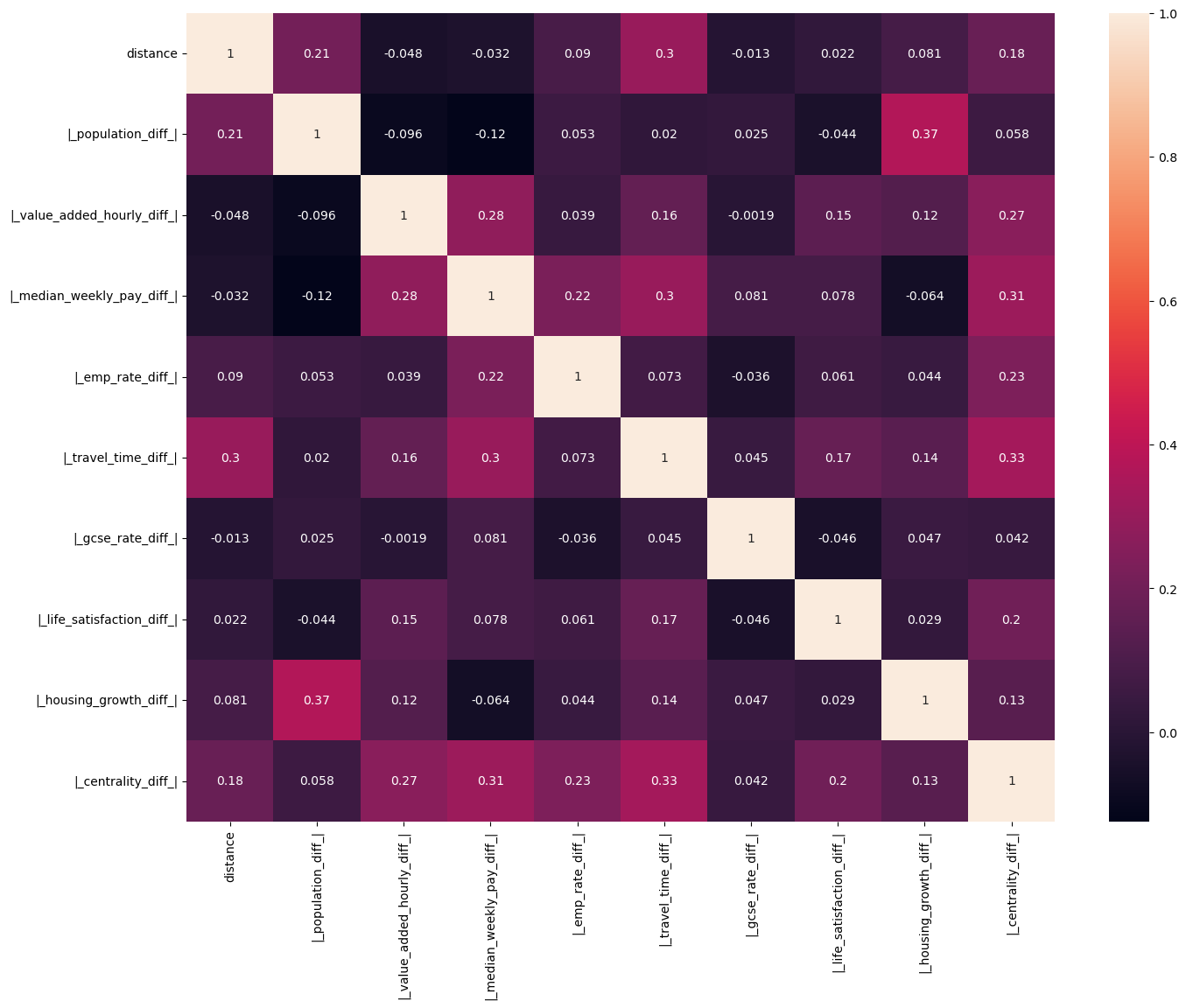
|  | Variable | Permutation Importance |
| --- | --- | --- |
| 0 | distance | 0.871311 |
| 1 | |\_population\_diff\_| | 0.038663 |
| 3 | |\_median\_weekly\_pay\_diff\_| | 0.020285 |
| 8 | |\_housing\_growth\_diff\_| | 0.017100 |

The results corroborate the Refined OLS with the four most influential variables and are again broadly consistent with the order.

Finally, with the dropped variable and transformed data, we can once again check for multicollinearity:

### 3.2.2 Multicollinearity

commuteOLSrcm = commuteOLSr.drop(columns=['pairs', 'journey\_score', 'route\_midpoint\_(geo)'])  
  
plt.figure(figsize=(16, 12))   
sns.heatmap(commuteOLSrcm.corr(), annot=True)



vif = pd.DataFrame()  
vif['Variable'] = X.columns  
vif['VIF'] = [variance\_inflation\_factor(X.values, i) for i in range(X.shape[1])]  
  
vif

|  | Variable | VIF |
| --- | --- | --- |
| 0 | const | 1.000000 |
| 1 | distance | 1.200264 |
| 2 | |\_population\_diff\_| | 1.243007 |
| 3 | |\_value\_added\_hourly\_diff\_| | 1.188374 |
| 4 | |\_median\_weekly\_pay\_diff\_| | 1.338856 |
| 5 | |\_emp\_rate\_diff\_| | 1.108864 |
| 6 | |\_travel\_time\_diff\_| | 1.348697 |
| 7 | |\_gcse\_rate\_diff\_| | 1.018920 |
| 8 | |\_life\_satisfaction\_diff\_| | 1.072590 |
| 9 | |\_housing\_growth\_diff\_| | 1.233093 |
| 10 | |\_centrality\_diff\_| | 1.333514 |

Where the concerns around multicollinearity are reduced.

### 3.2.3 Discussion

Generally the results are similar to the previous OLS:

* The top three variables in the new results are all within the top five of the previous OLS
  + The addition of 'median\_weekly\_pay\_diff' is consistent with previous results due to its correlation with 'avg\_monthly\_rent\_diff' which was dropped
    - Interetestingly the signs are different, suggesting as the differnce in median weekly pay increases, the 'journey\_score' decreases
* The order of influence/importance is the same across both the Simple and Refined OLS for the shared variables
* 'distance' is materially ahead of the other variables for both, and is the only variable with a coefficient above 0.1 in either model

With the transformation, 'distance' is an even more important predictor of 'journey\_score', with a larger coefficient (0.2728, +62.7% relative to simple OLS) and a more precise estimate. As intuitive as this is, it still must be critically challenged. This had been partially addressed by using absolute differences in other variables so they are directionless too.

What remains unconsidered is the potential for spatial autocorrelation, which is a key concern in geospatial data. A disproportionate amount of data points near and within London is observed in the [Figure 2](#fig-geo) produced in [Appendix B: Section 6.2](#sec-eda). Here, distances are shorter, and commuting flows more popular, which could inflate the role of 'distance' while underestimating the influence of other socio-economic structure variables. In dense areas like London, shorter distances are still popular, however this may not be because they are short, but due to other factors, e.g. high job density. In addition, socio-economic factors are likely to be more similar nearby, therefore, OLS may not be able to separate the two, instead attributing the high flows to distance alone. Given the potential sampling bias with the overrepresentation of structurally similar regions located near each other, these issues may be magnified as those relationships, in turn, dominate the model. Therefore, addressing some elements of spatial autocorrelation may be necessary.

## 3.3 Spatial Error Model (SEM)

I use spatial models (e.g., Spatial Error or Lag Models) where residuals may be spatially autocorrelated, as otherwise, the OLS assumption of independence of errors is violated, and inference restricted. Spatial models can account for spatial structure within the residuals and, therefore, if autocorrelation is present, are more reliable than a standard OLS regression. Clustering in commuting flows relating to types of roles has been discussed in the literature (Hincks et al., 2018), and it is reasonable to assume that this may also be present in the data, impacting other variables. Therefore, this will give me more confidence in the inference I can draw to answer the research question.

|  |
| --- |
| Further Information on Spatial Error Models (SEM) |
| The form of the model is (Harris and Jarvis, 2011):  Where:   * is the dependent variable * are the independent variables * is the intercept * are the coefficients * is the spatial autocorrelation coefficient * is the spatial weights matrix (i.e defines how observations are spatially connected) * is the spatially correlated error term * is the error term for errors not spatially correlated   Key Assumptions Specific to SEM:   * Spatial autocorrelation in residuals: The error term shows a spatial relationship, captured by . This can be tested for using Moran’s I * Correct specification of spatial weights: The matrix must should accurately reflect the true spatial structure (e.g. distance-based vs kNN) * No omitted spatial structure: All spatial dependence is captured within the relevant error term   All other assumptions (e.g. linearity, no perfect multicollinearity, homoscedasticity) are the same as in an OLS regression |

The SEM is a generalisation of the OLS model, and therefore, the same assumptions apply. The SEM also includes an additional assumption that the error term is spatially autocorrelated, which is the key difference between the two models.

I ran two spatial weight specification types (distance-based and k-nearest neighbours (kNN)) at various thresholds to mimic sensitivity analysis.

Distance-based weights define neighbours by the physical distance for a given threshold, and kNN defines neighbours by the number of nearest neighbours:

commuteSEM['residuals'] = rlr\_model.resid  
  
d\_thresholds = [10000, 20000, 50000, 100000, 200000] # in m by default  
k = [1, 2, 5, 10, 20]  
results = {'Distance-based': [], 'kNN': []}  
  
warnings.filterwarnings(  
 'ignore', message="The weights matrix is not fully connected\*") # get rid of error messags but can discuss  
  
# distance-based   
for d in d\_thresholds:  
 w = DistanceBand.from\_dataframe(  
 commuteSEM, threshold=d, silence\_warnings=True)  
 w.transform = 'r'  
 morani = Moran(commuteSEM['residuals'], w)  
 results['Distance-based'].append((d / 1000, morani.I, morani.p\_sim)) # switching to km  
  
# kNN  
for k in k:  
 w = KNN.from\_dataframe(commuteSEM, k=k)  
 w.transform = 'r'  
 morani = Moran(commuteSEM['residuals'], w)  
 results['kNN'].append((k, morani.I, morani.p\_sim))  
  
distance = pd.DataFrame(results['Distance-based'],  
 columns=['Distance (km)', "Moran's I", 'p-value'])  
knn = pd.DataFrame(results['kNN'], columns=['k', "Moran's I", 'p-value'])  
  
print("Moran's I (distance-based):")  
print(distance)  
print("\nMoran's I (kNN):")  
print(knn)

Moran's I (distance-based):  
 Distance (km) Moran's I p-value  
0 10.0 0.122814 0.001  
1 20.0 0.118623 0.001  
2 50.0 0.105237 0.001  
3 100.0 0.072716 0.001  
4 200.0 0.035270 0.001  
  
Moran's I (kNN):  
 k Moran's I p-value  
0 1 0.065393 0.088  
1 2 0.118213 0.001  
2 5 0.135020 0.001  
3 10 0.156095 0.001  
4 20 0.143852 0.001

I selected the kNN specification with 10 neighbours. At 0.156, the Moran’s I value suggests signficant spatial autocorrelation, and is the highest value. The distance-based method adds neighbours in polynomial (or near polynomial) form, leading to greater sensitivity in Moran I’s values. In addition, at most thresholds, not all points are connected which would result in an exclusion of remote data points and therefore bias the results. The kNN method is less sensitive to this, as it will always include the same number of neighbours regardless of distance and generally the kNN method is more interpretable, as it is easier to understand the number of neighbours than the distance between them.

The selection between a Spatial Error Model (SEM) and e.g. a Spatial Lag Model (SLM) is based on the nature of the spatial autocorrelation i.e. SEM is when the spatial autocorrelation is in the residuals, and SLM when in the dependent variable. To test this I use two Lagrange Multiplier (LM) tests respectively:

w = KNN.from\_dataframe(commuteSEM, k=10)  
w.transform = 'r'  
ols\_model = OLS(y, X, w=w, name\_y='journey\_score', name\_x=X.columns.tolist(), name\_w='kNN10')  
residuals = ols\_model.u   
  
# LM Error  
slm\_res = lag\_spatial(w, residuals)  
n = len(y)  
e = residuals  
e\_lag = slm\_res  
sse = np.sum(e\*\*2)  
sse\_lag = np.sum(e \* e\_lag)  
lm\_error = (n \* (sse\_lag\*\*2)) / (sse\*\*2)  
lm\_error\_p = 1 - stats.chi2.cdf(lm\_error, df=1)  
  
# LM lag  
y\_lag = lag\_spatial(w, y)  
lm\_lag = (n \* (np.sum(y\_lag \* e)\*\*2)) / (sse \* np.sum(y\_lag\*\*2))  
lm\_lag\_p = 1 - stats.chi2.cdf(lm\_lag, df=1)  
  
print(f"LM Error test: {lm\_error:.3f}, p-value: {lm\_error\_p:.3f}")  
print(f"LM Lag test: {lm\_lag:.3f}, p-value: {lm\_lag\_p:.3f}")

LM Error test: 16.496, p-value: 0.000  
LM Lag test: 0.000, p-value: 1.000

The tests imply that spatial autocorrelation exists in the residuals but not dependent variables, therefore a SEM is appropriate:

sem\_model = GM\_Error\_Het(y, X, w=w, name\_ds='commuteOLS' , name\_w="kNN, k=10", hard\_bound=True) # SEM with heteroskedasticity  
  
print(sem\_model.summary)

REGRESSION RESULTS  
------------------  
  
SUMMARY OF OUTPUT: GM SPATIALLY WEIGHTED LEAST SQUARES (HET)  
------------------------------------------------------------  
Data set : commuteOLS  
Weights matrix : kNN, k=10  
Dependent Variable :journey\_score Number of Observations: 677  
Mean dependent var : 0.3031 Number of Variables : 11  
S.D. dependent var : 0.4122 Degrees of Freedom : 666  
Pseudo R-squared : 0.4137  
N. of iterations : 1 Step1c computed : No  
  
------------------------------------------------------------------------------------  
 Variable Coefficient Std.Error z-Statistic Probability  
------------------------------------------------------------------------------------  
 CONSTANT 0.31282 0.03323 9.41436 0.00000  
 distance -0.34893 0.01441 -24.20732 0.00000  
 |\_population\_diff\_| 0.03319 0.01295 2.56337 0.01037  
|\_value\_added\_hourly\_diff\_| -0.00456 0.01158 -0.39342 0.69401  
|\_median\_weekly\_pay\_diff\_| -0.01664 0.01283 -1.29742 0.19449  
 |\_emp\_rate\_diff\_| 0.01863 0.01113 1.67371 0.09419  
|\_travel\_time\_diff\_| 0.01954 0.01451 1.34692 0.17801  
 |\_gcse\_rate\_diff\_| -0.01647 0.01027 -1.60488 0.10852  
|\_life\_satisfaction\_diff\_| 0.00155 0.01157 0.13438 0.89310  
|\_housing\_growth\_diff\_| 0.03430 0.01250 2.74306 0.00609  
 |\_centrality\_diff\_| 0.02457 0.01331 1.84668 0.06479  
 lambda 0.68295 0.03580 19.07529 0.00000  
------------------------------------------------------------------------------------  
Warning: Variable(s) ['const'] removed for being constant.  
================================ END OF REPORT =====================================

### 3.3.1 Interpretation of the SEM results

General SEM considerations:

* (Pseudo) R-squared = 0.4137
  + This metric is not directly comparable to R-squared, but shows a goodness of fit relative to a model with no variables
  + At 0.4137, this is a reasonable fit, explaining a fair amount of variation

Key Variable:

* 'distance', is again the most influential with the largest coefficient (0.34893), continuing to dominate in terms of importance even after the consideration of spatial autcorrelation
  + It is statistically significant still (p < 0.001)

Other variables:

* With the new model specification, 'median\_weekly\_pay\_diff' is no longer significant at any confidence level
* From the OLS summary, the significant variables ranked (by absolute size of coefficient) are as follows:

| # | Variable | Coefficient (β) | p-value | Interpretation vs Refined OLS |
| --- | --- | --- | --- | --- |
| 1 | distance | -0.3489 | 0.000 | Stronger negative effect |
| 2 | population\_diff | +0.0332 | 0.000 | Weaker positive effect |
| 3 | housing\_growth\_diff | +0.0343 | 0.006 | Weaker positive effect |
| - | All others | - | >0.05 | Not statistically significant at 95% confidence level |

The hypothesis that spatial autocorrelation prevented the OLS from seperating the impact of each vraiable driving the relationship was correct. However the direction for 'distance' was wrong, in actuality the other variables were soaking up the explanatory power of 'distance', instead of the other way around. Therefore, the SEM actually suggests that 'distance' is even more important than the OLS.

The exclusion of 'median\_weekly\_pay\_diff' is interesting, as it was significant in the OLS, but with a counterintuitive negative relationship between difference and flow popularity. This may have been due to spatial autocorrelation, proving further evidence that the SEM adds insight to the inference task.

This is about as deep as I can go with linear regression, and even then, it is not clear whether the SEM/OLS are robust enough, in terms of satisfying key assumptions, to draw an inference with confidence.

This can be seen when assessing the residuals of the SEM:

sns.set\_theme(style='darkgrid')  
palette = ['#69b3a2']  
  
plt.figure(figsize=(8, 6))  
sns.scatterplot(x=sem\_model.predy.flatten(),  
 y=sem\_model.u.flatten(), color=palette[0])  
plt.axhline(0, color='red', linestyle='--')  
plt.xlabel('Fitted values')  
plt.ylabel('Residuals')  
plt.title('Residuals vs Fitted')  
plt.tight\_layout()  
plt.show()

|  |
| --- |
| Figure 1: Scatterplot of residuals vs fitted values |

Here, we can see that despite best efforts, the variance of residuals changes with fitted values, i.e. heteroscedasticity persists.

# 4. Random Forest

Given that the assumptions of linear regression may not be fully satisfied in this context, I decide to use a Random Forest model which may be more appropriate.

Random Forests are more robust for inference as they do not rely on satisfying assumptions like linearity, homoscedasticity or independence of residuals. They handle outliers, multicollinearity, and non-linear relationships well, all of which may be relevant in this case.

The trade-off is that they do not provide effect sizes or directions when assessing feature importance directly, however in combination with the results of the regression models, they can provide a more complete picture of the relationships between variables.

This is shown below, using the original dataset (i.e. not standardised or transformed):

y\_rf = commute['journey\_score']  
X\_rf = commute[['distance', '|\_population\_diff\_|', '|\_value\_added\_hourly\_diff\_|',  
 '|\_median\_weekly\_pay\_diff\_|', '|\_emp\_rate\_diff\_|',  
 '|\_travel\_time\_diff\_|', '|\_gcse\_rate\_diff\_|',  
 '|\_life\_satisfaction\_diff\_|', '|\_housing\_growth\_diff\_|',  
 '|\_centrality\_diff\_|']]  
  
(X\_train,  
 X\_test,  
 y\_train,  
 y\_test) = skm.train\_test\_split(X\_rf,  
 y\_rf,  
 test\_size=0.2,  
 random\_state=0)  
  
commuteRF = RF(max\_features='sqrt',  
 random\_state=0).fit(X\_train, y\_train)  
y\_hat\_RF = commuteRF.predict(X\_test)  
mse = np.mean((y\_test - y\_hat\_RF)\*\*2)  
r2 = r2\_score(y\_test, y\_hat\_RF)  
  
print(f"Mean Squared Error: {mse.round(3)}")  
print(f"R-squared: {r2.round(3)}")

Mean Squared Error: 0.084  
R-squared: 0.485

The R-squared of the Random Forest model is 0.485, which is a marginal improvement on the linear regression models, implying that the random forest is able to better capture the relationships between the variables

Rather than coefficients, the Random Forest model has a feature importance metric, which is a measure of how much each variable contributes to the model’s predictions:

rif\_feature\_importance = pd.Series(commuteRF.feature\_importances\_, index=X\_train.columns).sort\_values(ascending=False)  
rif\_feature\_importance

distance 0.416896  
|\_population\_diff\_| 0.088291  
|\_housing\_growth\_diff\_| 0.074278  
|\_travel\_time\_diff\_| 0.073151  
|\_median\_weekly\_pay\_diff\_| 0.068869  
|\_centrality\_diff\_| 0.065763  
|\_value\_added\_hourly\_diff\_| 0.058756  
|\_gcse\_rate\_diff\_| 0.054705  
|\_emp\_rate\_diff\_| 0.051680  
|\_life\_satisfaction\_diff\_| 0.047610  
dtype: float64

Here, we can see that the most important variable is 'distance' (0.4169), which is consistent with the OLS and SEM results. The other variables again include 'population\_diff' (0.0883) and 'housing\_growth\_diff' (0.0743), consistent with the OLS and SEM results. However, 'travel\_time\_diff' (0.0732) is now also included, and 'median\_weekly\_pay\_diff' (0.0689) is not materially more important than the added 'centrality' variable (0.0658).

As discussed, Random Forest models are non-parametric, and hence, there is no additional information on concepts such as statistical significance, effect size, or direction of the relationship, in the scope of this method. Therefore, this is the extent of the inference.

And finally, permutation importance:

result = permutation\_importance(commuteRF, X\_test, y\_test,  
 n\_repeats=5,  
 random\_state=0)  
  
rif\_perm\_importance = pd.Series(result.importances\_mean, index=X\_test.columns).sort\_values(ascending=False)  
rif\_perm\_importance

distance 0.680163  
|\_population\_diff\_| 0.156520  
|\_centrality\_diff\_| 0.038961  
|\_median\_weekly\_pay\_diff\_| 0.019008  
|\_emp\_rate\_diff\_| 0.015234  
|\_travel\_time\_diff\_| 0.012933  
|\_housing\_growth\_diff\_| 0.011483  
|\_gcse\_rate\_diff\_| 0.010611  
|\_life\_satisfaction\_diff\_| -0.000230  
|\_value\_added\_hourly\_diff\_| -0.000486  
dtype: float64

The results are broadly consistent again, but 'centrality' is more important than other metrics for the first time. Again, it is important to note that there is an element of randomness in the results, and therefore, they must be interpreted with caution.

# 5. Conclusion

My research question asked:

“Which structural socioeconomic factors between regions most influence the relative popularity of commuting routes?”

I employed three models to address this question: OLS, SEM, and Random Forest. I used linear regressions as my primary method of analysis as the models tend to have the most inferential power, providing insight into importance, direction and effect size. However, the assumptions of linear regression may not be fully satisfied in this context, and therefore, I also employed a Random Forest model, which is more robust to violations of the assumptions. Other models were not used, as generally there is a trade-off between complexity and interpretability.

They key metrics used for inference were the coefficients of the OLS and SEM models, and the feature importance of the Random Forest model. I also used permutation importance across all, as it is a consistent metric across models, however due to the random elements, the confidence in their reults is lower. Other potential metrics such as Shapley values, were briefly mentioned, but not employed for simplicity of the task.

Results were generally consistent across the models. With high confidence, I can say that 'distance' is the most important structural variable that determines the relative popularity of commuting routes, and this is consistent with the literature on the subject (So et al., 2001). This was verified across all model types and specifications, and in each case, 'distance' was materially more important than the next best variable. In all regression models, the coefficient of 'distance' was negative, indicating that as distance increases, the popularity of the route decreases, and the results were statistically significant at all confidence levels. However, the literature (So et al., 2001) also focuses heavily on commuting time, which is not included in this analysis and is likely to be heavily correlated with distance. This is a key limitation of the analysis, as it is not possible to separate variables like 'distance' from related factors, and therefore inference is limited abnd/or biased by researcher design.

'population\_diff' was consistently the second most important variable across models (excluding the simple, pre-transformation OLS), and was statistically significant in all OLS and SEM models. These models showed that the larger the difference in the population, the higher the commuting flow’s relative popularity. This may reflect that larger populations by definiton have more people who can commute, leading to a higher commuting flow. It also may act as a proxy for other factors such as urban/rural splits, job density and many other socioeconomic factors.

Other consistently important variables included 'housing\_growth\_diff' and 'median\_weekly\_pay\_diff', where again, the greater the difference, the higher the relative popularity of the commuting flow. Importance rankings and significance varied across models, adding uncertainty to their interpretation. The former could be a lagging indicator of demand and, therefore, may be a proxy for other factors. The latter may reflect the economic trade-off associated with commuting.

Other variables did not consistently provide explanatory power for the relationship with commuting popularity, so I cannot draw any conclusions about their importance.

## References

Ahrens, A., & Lyons, S. (2021). Do rising rents lead to longer commutes? A gravity model of commuting flows in Ireland. Urban Studies, 58(2), 264–279. <https://doi.org/10.1177/0042098020910698>

Buckmann, M., & Joseph, A. (2022). An Interpretable Machine Learning Workflow with An Application to Economic Forecasting. SSRN Electronic Journal. <https://doi.org/10.2139/ssrn.4130517>

Harris, R., Jarvis, C. (2011). Statistics for Geography and Environmental Science (First Edition). Prentice Hall.

Hincks, S., Kingston, R., Webb, B., & Wong, C. (2018). A new geodemographic classification of commuting flows for England and Wales. International Journal of Geographical Information Science, 32(4), 663–684. <https://doi.org/10.1080/13658816.2017.1407416>

So, K. S., Orazem, P. F., & Otto, D. M. (2001). The Effects of Housing Prices, Wages, and Commuting Time on Joint Residential and Job Location Choices. American Journal of Agricultural Economics, 83(4), 1036–1048. <https://doi.org/10.1111/0002-9092.00228>

Tranos, E., Gheasi, M., & Nijkamp, P. (2015). International Migration: A Global Complex Network. Environment and Planning B: Planning and Design, 42(1), 4–22. <https://doi.org/10.1068/b39042>

Wander, M., & Blumenberg, E. (2024). Is housing affordability associated with shorter commutes for low-income workers? <https://doi.org/10.17610/T6Z32J>

Yeo, I.-K., & Johnson, R. (2000). A new family of power transformations to improve normality or symmetry. Biometrika, 87(4), 954–959. <https://doi.org/10.1093/biomet/87.4.954>

# 6. Appendices

## 6.1 Appendix A: Setup

### 6.1.1 Environment

This notebook was written in Python 3.9.18. The environment can be retrieved from the [Github project repo](https://github.com/2601547/commute).

### 6.1.2 Packages

import os  
import random  
import warnings  
  
import contextily as ctx  
import geopandas as gpd  
import matplotlib.pyplot as plt  
import networkx as nx  
import numpy as np  
import pandas as pd  
import requests  
import seaborn as sns  
import sklearn.model\_selection as skm  
import statsmodels.api as sm  
  
from esda.moran import Moran  
from libpysal.weights import DistanceBand, KNN, lag\_spatial  
from scipy import stats  
from scipy.stats import skew  
from sklearn.ensemble import RandomForestRegressor as RF  
from sklearn.inspection import permutation\_importance  
from sklearn.metrics import r2\_score  
from sklearn.preprocessing import PowerTransformer, StandardScaler  
from spreg import GM\_Error\_Het, OLS  
from statsmodels.stats.outliers\_influence import variance\_inflation\_factor

## 6.2 Appendix B: Exploratory Data Analysis

|  |
| --- |
| Section Summary |
| * Cleaned the dataset by:   + Dropping routes missing values (“City of London” and “Isles of Scilly”) * Add geographic information, including coordinates, using geopandas   + Identified potential spatial autocorrelation * Identified daily variation in 'journey\_score' for the same routes over time |

The following section addresses exploration of the provided dataset, focusing on:

* Understanding the data
* Ensuring the data is clean and usable
* Visualising the data to understand the context and distribution

The LAcommute.csv file is loaded into a pandas dataframe and displayed:

# relative file paths are used here as to not leak my name, although absolute paths may be better for reproducibility  
base\_data\_dir = os.path.abspath('../../1. Data/LAcommute')  
LAcommute = pd.read\_csv(  
 os.path.join(base\_data\_dir, "LAcommute.csv"), index\_col=0)  
LAcommute

|  | date | area\_name\_origin | area\_code\_origin | area\_name\_dest | area\_code\_dest | journey\_score | journey\_count\_decile | distance | population\_origin | population\_dest | ... | gcse\_rate\_origin | life\_satisfaction\_origin | housing\_growth\_origin | value\_added\_hourly\_dest | median\_weekly\_pay\_dest | emp\_rate\_dest | travel\_time\_dest | gcse\_rate\_dest | life\_satisfaction\_dest | housing\_growth\_dest |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 2019-01-01 | Hartlepool | E06000001 | Hartlepool | E06000001 | 1.4414 | 9 | 0.000000 | 92401 | 92401 | ... | 67.6 | 7.33 | 161 | 28.31 | 487.40 | 67.1 | 12.9 | 67.6 | 7.33 | 161 |
| 1 | 2019-01-01 | Hartlepool | E06000001 | County Durham | E06000047 | -0.3129 | 3 | 37592.170378 | 92401 | 518562 | ... | 67.6 | 7.33 | 161 | 28.96 | 469.40 | 71.4 | 14.1 | 67.6 | 7.43 | 1343 |
| 2 | 2019-01-01 | Middlesbrough | E06000002 | Middlesbrough | E06000002 | 1.0253 | 10 | 0.000000 | 142134 | 142134 | ... | 63.2 | 7.21 | 456 | 29.30 | 420.80 | 65.6 | 15.4 | 63.2 | 7.21 | 456 |
| 3 | 2019-01-01 | Middlesbrough | E06000002 | Redcar and Cleveland | E06000003 | 0.3086 | 7 | 13069.176565 | 142134 | 136699 | ... | 63.2 | 7.21 | 456 | 26.54 | 439.20 | 68.4 | 13.3 | 69.6 | 7.44 | 365 |
| 4 | 2019-01-01 | Middlesbrough | E06000002 | Stockton-on-Tees | E06000004 | 0.3772 | 8 | 7379.212731 | 142134 | 196860 | ... | 63.2 | 7.21 | 456 | 34.37 | 469.40 | 74.8 | 13.2 | 69.5 | 7.40 | 616 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 316514 | 2019-12-31 | Westminster | E09000033 | Sutton | E09000029 | -0.6732 | 4 | 16964.439602 | 208415 | 208516 | ... | 77.3 | 7.21 | 580 | 35.19 | 565.80 | 77.4 | 9.0 | 82.0 | 7.36 | 313 |
| 316515 | 2019-12-31 | Westminster | E09000033 | Tower Hamlets | E09000030 | 1.3720 | 10 | 8616.142460 | 208415 | 305066 | ... | 77.3 | 7.21 | 580 | 60.46 | 680.30 | 74.4 | 4.4 | 72.4 | 7.13 | 3248 |
| 316516 | 2019-12-31 | Westminster | E09000033 | Waltham Forest | E09000031 | -0.4483 | 6 | 13672.865893 | 208415 | 281015 | ... | 77.3 | 7.21 | 580 | 34.63 | 624.70 | 71.5 | 7.2 | 71.5 | 7.30 | 1263 |
| 316517 | 2019-12-31 | Westminster | E09000033 | Wandsworth | E09000032 | 0.1871 | 8 | 7117.584240 | 208415 | 334558 | ... | 77.3 | 7.21 | 580 | 35.15 | 746.70 | 84.9 | 6.2 | 74.2 | 7.34 | 1415 |
| 316518 | 2019-12-31 | Westminster | E09000033 | Westminster | E09000033 | 1.6534 | 10 | 0.000000 | 208415 | 208415 | ... | 77.3 | 7.21 | 580 | 52.46 | 771.60 | 67.2 | 5.1 | 77.3 | 7.21 | 580 |

The 316519 rows represent various commuting flow routes within and between different areas of the UK from 1 January 2019 to 31 December 2019. Routes are reported as pairs of areas, with the origin and destination columns indicating each route’s start and end points. Given the areas are reported as local authorities (LA), as per the area codes, and implied from the dataset title, this data is likely sourced from the Office for National Statistics (ONS).

The first row shows “Hartlepool” to “Hartlepool,” meaning there is both intra- and inter-LA commuting.

There are 24 columns, which are listed as:

LAcommute.columns

Index(['date', 'area\_name\_origin', 'area\_code\_origin', 'area\_name\_dest',  
 'area\_code\_dest', 'journey\_score', 'journey\_count\_decile', 'distance',  
 'population\_origin', 'population\_dest', 'value\_added\_hourly\_origin',  
 'median\_weekly\_pay\_origin', 'emp\_rate\_origin', 'travel\_time\_origin',  
 'gcse\_rate\_origin', 'life\_satisfaction\_origin', 'housing\_growth\_origin',  
 'value\_added\_hourly\_dest', 'median\_weekly\_pay\_dest', 'emp\_rate\_dest',  
 'travel\_time\_dest', 'gcse\_rate\_dest', 'life\_satisfaction\_dest',  
 'housing\_growth\_dest'],  
 dtype='object')

We can also see some additional information about the dataset using describe()

LAcommute.describe()

|  | journey\_score | journey\_count\_decile | distance | population\_origin | population\_dest | value\_added\_hourly\_origin | travel\_time\_origin | housing\_growth\_origin | value\_added\_hourly\_dest | travel\_time\_dest | housing\_growth\_dest |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| count | 316519.000000 | 316519.000000 | 316519.000000 | 3.165190e+05 | 3.165190e+05 | 316519.000000 | 316519.000000 | 316519.000000 | 316519.000000 | 316519.000000 | 316519.000000 |
| mean | 0.519814 | 6.280928 | 13044.446197 | 2.840599e+05 | 2.878074e+05 | 39.152417 | 8.476279 | 1158.507954 | 38.540809 | 8.655166 | 1160.971885 |
| std | 0.996988 | 2.821859 | 10767.620494 | 1.393700e+05 | 1.347657e+05 | 9.230682 | 2.772127 | 870.434423 | 8.907160 | 2.693593 | 852.007444 |
| min | -2.168800 | 1.000000 | 0.000000 | 2.098000e+03 | 2.098000e+03 | 23.790000 | 3.200000 | 0.000000 | 23.790000 | 3.200000 | 0.000000 |
| 25% | -0.230050 | 4.000000 | 6993.581679 | 2.084150e+05 | 2.122450e+05 | 31.820000 | 6.200000 | 547.000000 | 31.270000 | 7.000000 | 551.000000 |
| 50% | 0.464000 | 7.000000 | 11461.611507 | 2.789080e+05 | 2.810150e+05 | 36.820000 | 8.300000 | 969.000000 | 36.800000 | 8.400000 | 969.000000 |
| 75% | 1.172200 | 9.000000 | 17143.862769 | 3.311920e+05 | 3.345580e+05 | 46.140000 | 10.300000 | 1439.000000 | 45.240000 | 10.400000 | 1447.000000 |
| max | 8.677600 | 10.000000 | 269291.080357 | 1.150646e+06 | 1.150646e+06 | 60.810000 | 44.300000 | 4024.000000 | 60.810000 | 44.300000 | 4024.000000 |

We can see that 'journey\_score,' which will likely be the dependent variable, appears to have been standardised, i.e., a z-score is given for each pair. The mean is 0.5, close to the value of 0 you would expect, and the standard deviation is 1.0. The delta in the mean and 0 suggests that this may be a subset of a larger dataset that was standardised.

This also means that scores are relative, with scores of 0.5 and below reflecting a route that is less travelled than average and scores above 0.5 reflecting a more travelled route than average.

Further research identified that the original data source for this 'journey\_score' variable is from the [CDRC](https://data.cdrc.ac.uk/dataset/spectus-origin-destination-derived-mobility-data) based on mobile GPS data. The 'journey\_score' variable is confirmed as a z-score, standardised over 2019 to 2022. This presents a significant issue for interpretation as the interpretation becomes more akin to:

*“Holding other factors constant, a one-unit increase in the variable 'A' is associated with a β change in the journey\_score, measured in standard deviations relative to the 2019–2022 distribution”*

Which may not be intuitive to interpret or meaningful in the context of the research question.

More information can be found .info()

LAcommute.info()

<class 'pandas.core.frame.DataFrame'>  
Index: 316519 entries, 0 to 316518  
Data columns (total 24 columns):  
 # Column Non-Null Count Dtype   
--- ------ -------------- -----   
 0 date 316519 non-null object   
 1 area\_name\_origin 316519 non-null object   
 2 area\_code\_origin 316519 non-null object   
 3 area\_name\_dest 316519 non-null object   
 4 area\_code\_dest 316519 non-null object   
 5 journey\_score 316519 non-null float64  
 6 journey\_count\_decile 316519 non-null int64   
 7 distance 316519 non-null float64  
 8 population\_origin 316519 non-null int64   
 9 population\_dest 316519 non-null int64   
 10 value\_added\_hourly\_origin 316519 non-null float64  
 11 median\_weekly\_pay\_origin 316519 non-null object   
 12 emp\_rate\_origin 316519 non-null object   
 13 travel\_time\_origin 316519 non-null float64  
 14 gcse\_rate\_origin 316519 non-null object   
 15 life\_satisfaction\_origin 316519 non-null object   
 16 housing\_growth\_origin 316519 non-null int64   
 17 value\_added\_hourly\_dest 316519 non-null float64  
 18 median\_weekly\_pay\_dest 316519 non-null object   
 19 emp\_rate\_dest 316519 non-null object   
 20 travel\_time\_dest 316519 non-null float64  
 21 gcse\_rate\_dest 316519 non-null object   
 22 life\_satisfaction\_dest 316519 non-null object   
 23 housing\_growth\_dest 316519 non-null int64   
dtypes: float64(6), int64(5), object(13)  
memory usage: 60.4+ MB

We also see that the data is not completely usable as provided due to inconsistent data types. For example, the column 'median\_weekly\_pay\_origin' is an object type with the incorrect storage style. This can be amended:

LAcommute.dtypes

date object  
area\_name\_origin object  
area\_code\_origin object  
area\_name\_dest object  
area\_code\_dest object  
journey\_score float64  
journey\_count\_decile int64  
distance float64  
population\_origin int64  
population\_dest int64  
value\_added\_hourly\_origin float64  
median\_weekly\_pay\_origin object  
emp\_rate\_origin object  
travel\_time\_origin float64  
gcse\_rate\_origin object  
life\_satisfaction\_origin object  
housing\_growth\_origin int64  
value\_added\_hourly\_dest float64  
median\_weekly\_pay\_dest object  
emp\_rate\_dest object  
travel\_time\_dest float64  
gcse\_rate\_dest object  
life\_satisfaction\_dest object  
housing\_growth\_dest int64  
dtype: object

LAcommute['date'] = LAcommute['date'].apply(  
 pd.to\_datetime, format='%Y-%m-%d', errors='coerce')  
other\_columns = ['emp\_rate\_origin', 'emp\_rate\_dest',  
 'median\_weekly\_pay\_origin', 'median\_weekly\_pay\_dest',  
 'gcse\_rate\_origin', 'gcse\_rate\_dest',  
 'life\_satisfaction\_origin', 'life\_satisfaction\_dest',]  
LAcommute[other\_columns] = LAcommute[other\_columns].apply(  
 pd.to\_numeric, errors='coerce')  
  
LAcommute.dtypes

date datetime64[ns]  
area\_name\_origin object  
area\_code\_origin object  
area\_name\_dest object  
area\_code\_dest object  
journey\_score float64  
journey\_count\_decile int64  
distance float64  
population\_origin int64  
population\_dest int64  
value\_added\_hourly\_origin float64  
median\_weekly\_pay\_origin float64  
emp\_rate\_origin float64  
travel\_time\_origin float64  
gcse\_rate\_origin float64  
life\_satisfaction\_origin float64  
housing\_growth\_origin int64  
value\_added\_hourly\_dest float64  
median\_weekly\_pay\_dest float64  
emp\_rate\_dest float64  
travel\_time\_dest float64  
gcse\_rate\_dest float64  
life\_satisfaction\_dest float64  
housing\_growth\_dest int64  
dtype: object

We can then check for missing values and or duplicates:

if LAcommute.isnull().values.any():  
 print("Missing values found")  
else:  
 print("No missing values")  
  
if LAcommute[LAcommute.duplicated()].empty:  
 print("No duplicates")  
else:  
 print("Duplicates found")

Missing values found  
No duplicates

Although there are no duplicates, we can see some missing values. We can see the number of missing values in each column using:

missing\_values = LAcommute.isnull().sum()  
missing\_values

date 0  
area\_name\_origin 0  
area\_code\_origin 0  
area\_name\_dest 0  
area\_code\_dest 0  
journey\_score 0  
journey\_count\_decile 0  
distance 0  
population\_origin 0  
population\_dest 0  
value\_added\_hourly\_origin 0  
median\_weekly\_pay\_origin 9599  
emp\_rate\_origin 9599  
travel\_time\_origin 0  
gcse\_rate\_origin 9586  
life\_satisfaction\_origin 9599  
housing\_growth\_origin 0  
value\_added\_hourly\_dest 0  
median\_weekly\_pay\_dest 6271  
emp\_rate\_dest 6271  
travel\_time\_dest 0  
gcse\_rate\_dest 6258  
life\_satisfaction\_dest 6271  
housing\_growth\_dest 0  
dtype: int64

That is a significant amount of missing values, so it is important to determine what is going on and following that, how to deal with them. We can see that they are not random, they are only for 'median\_weekly\_pay', 'emp\_rate', 'gcse\_rate' and 'life\_satisfaction'.

missing\_values\_rows = LAcommute[LAcommute.isnull().any(axis=1)]  
missing\_values\_rows

|  | date | area\_name\_origin | area\_code\_origin | area\_name\_dest | area\_code\_dest | journey\_score | journey\_count\_decile | distance | population\_origin | population\_dest | ... | gcse\_rate\_origin | life\_satisfaction\_origin | housing\_growth\_origin | value\_added\_hourly\_dest | median\_weekly\_pay\_dest | emp\_rate\_dest | travel\_time\_dest | gcse\_rate\_dest | life\_satisfaction\_dest | housing\_growth\_dest |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 294 | 2019-01-01 | City of London | E09000001 | City of London | E09000001 | -0.3374 | 10 | 0.000000 | 8765 | 8765 | ... | NaN | NaN | 206 | 57.01 | NaN | NaN | 7.9 | NaN | NaN | 206 |
| 295 | 2019-01-01 | City of London | E09000001 | Barking and Dagenham | E09000002 | -1.3314 | 2 | 16110.974505 | 8765 | 218828 | ... | NaN | NaN | 206 | 36.89 | 523.5 | 67.3 | 8.4 | 68.0 | 7.35 | 1048 |
| 296 | 2019-01-01 | City of London | E09000001 | Brent | E09000005 | -1.1846 | 2 | 13051.679464 | 8765 | 347424 | ... | NaN | NaN | 206 | 36.82 | 553.1 | 70.4 | 7.5 | 71.5 | 7.25 | 2404 |
| 297 | 2019-01-01 | City of London | E09000001 | Camden | E09000007 | -0.7811 | 6 | 5850.378829 | 8765 | 217136 | ... | NaN | NaN | 206 | 51.32 | 694.2 | 69.6 | 5.7 | 72.3 | 6.78 | 509 |
| 298 | 2019-01-01 | City of London | E09000001 | Hackney | E09000012 | -0.8153 | 5 | 4665.516680 | 8765 | 265825 | ... | NaN | NaN | 206 | 36.06 | 575.1 | 72.5 | 4.8 | 72.0 | 6.94 | 969 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 316393 | 2019-12-31 | Newham | E09000025 | City of London | E09000001 | 0.3497 | 6 | 9090.106893 | 349786 | 8765 | ... | 70.7 | 7.51 | 1813 | 57.01 | NaN | NaN | 7.9 | NaN | NaN | 206 |
| 316422 | 2019-12-31 | Southwark | E09000028 | City of London | E09000001 | 1.4381 | 9 | 4597.455204 | 312591 | 8765 | ... | 71.0 | 7.17 | 1096 | 57.01 | NaN | NaN | 7.9 | NaN | NaN | 206 |
| 316448 | 2019-12-31 | Tower Hamlets | E09000030 | City of London | E09000001 | 0.5614 | 9 | 3961.057843 | 305066 | 8765 | ... | 72.4 | 7.13 | 3248 | 57.01 | NaN | NaN | 7.9 | NaN | NaN | 206 |
| 316464 | 2019-12-31 | Waltham Forest | E09000031 | City of London | E09000001 | -1.0617 | 1 | 10581.867402 | 281015 | 8765 | ... | 71.5 | 7.30 | 1263 | 57.01 | NaN | NaN | 7.9 | NaN | NaN | 206 |
| 316489 | 2019-12-31 | Westminster | E09000033 | City of London | E09000001 | 2.0080 | 10 | 4660.462296 | 208415 | 8765 | ... | 77.3 | 7.21 | 580 | 57.01 | NaN | NaN | 7.9 | NaN | NaN | 206 |

Since all rows appear to have the commonality of “City of London”, we can assume this is the source of the missing value. This is consistent with the implication that this is ONS data, as the City of London tends to be excluded from datasets due to its unique nature and small population.

It is simple to remove these rows, as they are likely irrelevant to the analysis.

LAcommute = LAcommute[  
 (LAcommute['area\_name\_origin'] != 'City of London') &  
 (LAcommute['area\_name\_dest'] != 'City of London')  
]

We can once again check for missing values…

missing\_values = LAcommute.isnull().sum()  
missing\_values

date 0  
area\_name\_origin 0  
area\_code\_origin 0  
area\_name\_dest 0  
area\_code\_dest 0  
journey\_score 0  
journey\_count\_decile 0  
distance 0  
population\_origin 0  
population\_dest 0  
value\_added\_hourly\_origin 0  
median\_weekly\_pay\_origin 13  
emp\_rate\_origin 13  
travel\_time\_origin 0  
gcse\_rate\_origin 0  
life\_satisfaction\_origin 13  
housing\_growth\_origin 0  
value\_added\_hourly\_dest 0  
median\_weekly\_pay\_dest 13  
emp\_rate\_dest 13  
travel\_time\_dest 0  
gcse\_rate\_dest 0  
life\_satisfaction\_dest 13  
housing\_growth\_dest 0  
dtype: int64

However it appears that there are still some missing values in the dataset.

missing\_values\_rows = LAcommute[LAcommute.isnull().any(axis=1)]  
missing\_values\_rows

|  | date | area\_name\_origin | area\_code\_origin | area\_name\_dest | area\_code\_dest | journey\_score | journey\_count\_decile | distance | population\_origin | population\_dest | ... | gcse\_rate\_origin | life\_satisfaction\_origin | housing\_growth\_origin | value\_added\_hourly\_dest | median\_weekly\_pay\_dest | emp\_rate\_dest | travel\_time\_dest | gcse\_rate\_dest | life\_satisfaction\_dest | housing\_growth\_dest |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 123116 | 2019-05-04 | Isles of Scilly | E06000053 | Isles of Scilly | E06000053 | 0.7176 | 2 | 0.0 | 2098 | 2098 | ... | 89.5 | NaN | 0 | 39.28 | NaN | NaN | 44.3 | 89.5 | NaN | 0 |
| 124824 | 2019-05-06 | Isles of Scilly | E06000053 | Isles of Scilly | E06000053 | -0.6279 | 1 | 0.0 | 2098 | 2098 | ... | 89.5 | NaN | 0 | 39.28 | NaN | NaN | 44.3 | 89.5 | NaN | 0 |
| 200539 | 2019-08-07 | Isles of Scilly | E06000053 | Isles of Scilly | E06000053 | 2.0631 | 2 | 0.0 | 2098 | 2098 | ... | 89.5 | NaN | 0 | 39.28 | NaN | NaN | 44.3 | 89.5 | NaN | 0 |
| 201407 | 2019-08-08 | Isles of Scilly | E06000053 | Isles of Scilly | E06000053 | 2.0631 | 2 | 0.0 | 2098 | 2098 | ... | 89.5 | NaN | 0 | 39.28 | NaN | NaN | 44.3 | 89.5 | NaN | 0 |
| 202255 | 2019-08-09 | Isles of Scilly | E06000053 | Isles of Scilly | E06000053 | -0.6279 | 1 | 0.0 | 2098 | 2098 | ... | 89.5 | NaN | 0 | 39.28 | NaN | NaN | 44.3 | 89.5 | NaN | 0 |
| 208769 | 2019-08-17 | Isles of Scilly | E06000053 | Isles of Scilly | E06000053 | -0.6279 | 1 | 0.0 | 2098 | 2098 | ... | 89.5 | NaN | 0 | 39.28 | NaN | NaN | 44.3 | 89.5 | NaN | 0 |
| 211118 | 2019-08-20 | Isles of Scilly | E06000053 | Isles of Scilly | E06000053 | 0.7176 | 2 | 0.0 | 2098 | 2098 | ... | 89.5 | NaN | 0 | 39.28 | NaN | NaN | 44.3 | 89.5 | NaN | 0 |
| 216123 | 2019-08-26 | Isles of Scilly | E06000053 | Isles of Scilly | E06000053 | -0.6279 | 1 | 0.0 | 2098 | 2098 | ... | 89.5 | NaN | 0 | 39.28 | NaN | NaN | 44.3 | 89.5 | NaN | 0 |
| 216842 | 2019-08-27 | Isles of Scilly | E06000053 | Isles of Scilly | E06000053 | -0.6279 | 1 | 0.0 | 2098 | 2098 | ... | 89.5 | NaN | 0 | 39.28 | NaN | NaN | 44.3 | 89.5 | NaN | 0 |
| 221053 | 2019-09-01 | Isles of Scilly | E06000053 | Isles of Scilly | E06000053 | -0.6279 | 1 | 0.0 | 2098 | 2098 | ... | 89.5 | NaN | 0 | 39.28 | NaN | NaN | 44.3 | 89.5 | NaN | 0 |
| 224334 | 2019-09-05 | Isles of Scilly | E06000053 | Isles of Scilly | E06000053 | 0.7176 | 2 | 0.0 | 2098 | 2098 | ... | 89.5 | NaN | 0 | 39.28 | NaN | NaN | 44.3 | 89.5 | NaN | 0 |
| 226828 | 2019-09-08 | Isles of Scilly | E06000053 | Isles of Scilly | E06000053 | -0.6279 | 1 | 0.0 | 2098 | 2098 | ... | 89.5 | NaN | 0 | 39.28 | NaN | NaN | 44.3 | 89.5 | NaN | 0 |
| 231037 | 2019-09-13 | Isles of Scilly | E06000053 | Isles of Scilly | E06000053 | -0.6279 | 1 | 0.0 | 2098 | 2098 | ... | 89.5 | NaN | 0 | 39.28 | NaN | NaN | 44.3 | 89.5 | NaN | 0 |

…which this time appears to be “Isles of Scilly”. We remove these too, noting it looks like all “Isles of Scilly” routes are to the “Isles of Scilly”.

LAcommute = LAcommute[  
 (LAcommute['area\_name\_origin'] != 'Isles of Scilly')  
]

date 0  
area\_name\_origin 0  
area\_code\_origin 0  
area\_name\_dest 0  
area\_code\_dest 0  
journey\_score 0  
journey\_count\_decile 0  
distance 0  
population\_origin 0  
population\_dest 0  
value\_added\_hourly\_origin 0  
median\_weekly\_pay\_origin 0  
emp\_rate\_origin 0  
travel\_time\_origin 0  
gcse\_rate\_origin 0  
life\_satisfaction\_origin 0  
housing\_growth\_origin 0  
value\_added\_hourly\_dest 0  
median\_weekly\_pay\_dest 0  
emp\_rate\_dest 0  
travel\_time\_dest 0  
gcse\_rate\_dest 0  
life\_satisfaction\_dest 0  
housing\_growth\_dest 0  
dtype: int64

It may be helpful to get a sense of where the routes are, as this will determine the scope of the analysis. First, we can list the unique local authorities in the dataset:

LA\_count = LAcommute['area\_name\_origin'].nunique()  
LA\_count

119

There are 119 LAs in the dataset, following the removal of the City of London and Isles of Scilly. To contexutalise we this can be plotted on map.

The coordinates for the LA can be found in the [government depo](https://geoportal.statistics.gov.uk/datasets/9cb3c710143649499ff6acaca927d205_0/explore), and is downloaded as the file "LAD\_Dec\_2019\_Boundaries\_UK\_BFC\_2022\_-5126023737554987305.csv".

LA\_coordinates = pd.read\_csv(  
 os.path.join(base\_data\_dir, "LAD\_Dec\_2019\_Boundaries\_UK\_BFC\_2022\_-5126023737554987305.csv"), index\_col=0)  
  
LA\_coordinates.head()

|  | objectid | lad19cd | lad19nm | lad19nmw | bng\_e | bng\_n | long | lat | st\_areasha | st\_lengths | Shape\_\_Area | Shape\_\_Length | GlobalID |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| FID |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 1 | 1 | E06000001 | Hartlepool |  | 447160 | 531474 | -1.27018 | 54.676140 | 9.371262e+07 | 71011.933949 | 2.797890e+08 | 122680.152623 | d3127426-63c8-4358-a554-b3fe4f0e537e |
| 2 | 2 | E06000002 | Middlesbrough |  | 451141 | 516887 | -1.21099 | 54.544670 | 5.388156e+07 | 44481.691242 | 1.598722e+08 | 76614.777246 | 7caac8a1-a10b-4878-a2ab-4fdb80e987a4 |
| 3 | 3 | E06000003 | Redcar and Cleveland |  | 464361 | 519597 | -1.00608 | 54.567520 | 2.450695e+08 | 96703.989701 | 7.274510e+08 | 166599.724875 | 96cc264b-6155-48cb-8fc0-7e41684d2682 |
| 4 | 4 | E06000004 | Stockton-on-Tees |  | 444940 | 518183 | -1.30664 | 54.556911 | 2.049330e+08 | 123408.985928 | 6.086362e+08 | 212698.045558 | 051ae100-c81d-4cc1-989b-b65e71a9b942 |
| 5 | 5 | E06000005 | Darlington |  | 428029 | 515648 | -1.56835 | 54.535339 | 1.974757e+08 | 107206.401677 | 5.861353e+08 | 184666.724022 | c22c366b-0133-4b9a-8376-013e0c43d6f2 |

The coordinates are then added to the dataframe…

def merge\_coordinates(df, coordinates, merge\_col, prefix):  
 df = df.merge(  
 coordinates[['lad19nm', 'lat', 'long']],  
 how='left',  
 left\_on=merge\_col,  
 right\_on='lad19nm'  
 )  
 df = df.rename(columns={  
 'lat': f'lat\_{prefix}',  
 'long': f'long\_{prefix}'  
 })  
 return df.drop(columns=['lad19nm'])

LAcommute = merge\_coordinates(LAcommute, LA\_coordinates, 'area\_name\_origin', 'origin')  
LAcommute = merge\_coordinates(LAcommute, LA\_coordinates, 'area\_name\_dest', 'dest')  
  
LAcommute

|  | date | area\_name\_origin | area\_code\_origin | area\_name\_dest | area\_code\_dest | journey\_score | journey\_count\_decile | distance | population\_origin | population\_dest | ... | median\_weekly\_pay\_dest | emp\_rate\_dest | travel\_time\_dest | gcse\_rate\_dest | life\_satisfaction\_dest | housing\_growth\_dest | lat\_origin | long\_origin | lat\_dest | long\_dest |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 2019-01-01 | Hartlepool | E06000001 | Hartlepool | E06000001 | 1.4414 | 9 | 0.000000 | 92401 | 92401 | ... | 487.4 | 67.1 | 12.9 | 67.6 | 7.33 | 161 | 54.676140 | -1.27018 | 54.676140 | -1.27018 |
| 1 | 2019-01-01 | Hartlepool | E06000001 | County Durham | E06000047 | -0.3129 | 3 | 37592.170378 | 92401 | 518562 | ... | 469.4 | 71.4 | 14.1 | 67.6 | 7.43 | 1343 | 54.676140 | -1.27018 | 54.685131 | -1.84050 |
| 2 | 2019-01-01 | Middlesbrough | E06000002 | Middlesbrough | E06000002 | 1.0253 | 10 | 0.000000 | 142134 | 142134 | ... | 420.8 | 65.6 | 15.4 | 63.2 | 7.21 | 456 | 54.544670 | -1.21099 | 54.544670 | -1.21099 |
| 3 | 2019-01-01 | Middlesbrough | E06000002 | Redcar and Cleveland | E06000003 | 0.3086 | 7 | 13069.176565 | 142134 | 136699 | ... | 439.2 | 68.4 | 13.3 | 69.6 | 7.44 | 365 | 54.544670 | -1.21099 | 54.567520 | -1.00608 |
| 4 | 2019-01-01 | Middlesbrough | E06000002 | Stockton-on-Tees | E06000004 | 0.3772 | 8 | 7379.212731 | 142134 | 196860 | ... | 469.4 | 74.8 | 13.2 | 69.5 | 7.40 | 616 | 54.544670 | -1.21099 | 54.556911 | -1.30664 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 301022 | 2019-12-31 | Westminster | E09000033 | Sutton | E09000029 | -0.6732 | 4 | 16964.439602 | 208415 | 208516 | ... | 565.8 | 77.4 | 9.0 | 82.0 | 7.36 | 313 | 51.512199 | -0.15295 | 51.357559 | -0.17227 |
| 301023 | 2019-12-31 | Westminster | E09000033 | Tower Hamlets | E09000030 | 1.3720 | 10 | 8616.142460 | 208415 | 305066 | ... | 680.3 | 74.4 | 4.4 | 72.4 | 7.13 | 3248 | 51.512199 | -0.15295 | 51.515541 | -0.03643 |
| 301024 | 2019-12-31 | Westminster | E09000033 | Waltham Forest | E09000031 | -0.4483 | 6 | 13672.865893 | 208415 | 281015 | ... | 624.7 | 71.5 | 7.2 | 71.5 | 7.30 | 1263 | 51.512199 | -0.15295 | 51.594608 | -0.01881 |
| 301025 | 2019-12-31 | Westminster | E09000033 | Wandsworth | E09000032 | 0.1871 | 8 | 7117.584240 | 208415 | 334558 | ... | 746.7 | 84.9 | 6.2 | 74.2 | 7.34 | 1415 | 51.512199 | -0.15295 | 51.452400 | -0.20023 |
| 301026 | 2019-12-31 | Westminster | E09000033 | Westminster | E09000033 | 1.6534 | 10 | 0.000000 | 208415 | 208415 | ... | 771.6 | 67.2 | 5.1 | 77.3 | 7.21 | 580 | 51.512199 | -0.15295 | 51.512199 | -0.15295 |

And then the dataframe is converted to a geodataframe:

LAcommute\_geo = LAcommute # a gdf with two geometries is difficult to save, so for the appendices I will use this temporarily and in the main recreate a seperate LAcommuute gdf  
  
LAcommute\_geo['geometry\_origin'] = gpd.points\_from\_xy(  
 LAcommute\_geo['long\_origin'], LAcommute\_geo['lat\_origin']  
)  
LAcommute\_geo['geometry\_dest'] = gpd.points\_from\_xy(  
 LAcommute\_geo['long\_dest'], LAcommute\_geo['lat\_dest']  
)  
  
LAcommute\_geo = gpd.GeoDataFrame(  
 LAcommute\_geo,  
 geometry='geometry\_origin', # although I have added both geometries, for simlplicity I will only use the origin geometry for now  
 crs='EPSG:4326'  
)  
  
LAcommute\_geo = LAcommute\_geo.drop(columns=['lat\_origin', 'long\_origin', 'lat\_dest', 'long\_dest'])  
LAcommute\_geo.head()

|  | date | area\_name\_origin | area\_code\_origin | area\_name\_dest | area\_code\_dest | journey\_score | journey\_count\_decile | distance | population\_origin | population\_dest | ... | housing\_growth\_origin | value\_added\_hourly\_dest | median\_weekly\_pay\_dest | emp\_rate\_dest | travel\_time\_dest | gcse\_rate\_dest | life\_satisfaction\_dest | housing\_growth\_dest | geometry\_origin | geometry\_dest |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 2019-01-01 | Hartlepool | E06000001 | Hartlepool | E06000001 | 1.4414 | 9 | 0.000000 | 92401 | 92401 | ... | 161 | 28.31 | 487.4 | 67.1 | 12.9 | 67.6 | 7.33 | 161 | POINT (-1.27018 54.67614) | POINT (-1.27018 54.67614) |
| 1 | 2019-01-01 | Hartlepool | E06000001 | County Durham | E06000047 | -0.3129 | 3 | 37592.170378 | 92401 | 518562 | ... | 161 | 28.96 | 469.4 | 71.4 | 14.1 | 67.6 | 7.43 | 1343 | POINT (-1.27018 54.67614) | POINT (-1.84050 54.68513) |
| 2 | 2019-01-01 | Middlesbrough | E06000002 | Middlesbrough | E06000002 | 1.0253 | 10 | 0.000000 | 142134 | 142134 | ... | 456 | 29.30 | 420.8 | 65.6 | 15.4 | 63.2 | 7.21 | 456 | POINT (-1.21099 54.54467) | POINT (-1.21099 54.54467) |
| 3 | 2019-01-01 | Middlesbrough | E06000002 | Redcar and Cleveland | E06000003 | 0.3086 | 7 | 13069.176565 | 142134 | 136699 | ... | 456 | 26.54 | 439.2 | 68.4 | 13.3 | 69.6 | 7.44 | 365 | POINT (-1.21099 54.54467) | POINT (-1.00608 54.56752) |
| 4 | 2019-01-01 | Middlesbrough | E06000002 | Stockton-on-Tees | E06000004 | 0.3772 | 8 | 7379.212731 | 142134 | 196860 | ... | 456 | 34.37 | 469.4 | 74.8 | 13.2 | 69.5 | 7.40 | 616 | POINT (-1.21099 54.54467) | POINT (-1.30664 54.55691) |

and now this can be plotted on a map:

# the is reprojected (from EPSG:4326) to EPSG:3857  
LAcommute\_geo = LAcommute\_geo.to\_crs(epsg=3857)  
  
fig, ax = plt.subplots(figsize=(14, 18))  
  
LAcommute\_geo.plot(ax=ax, alpha=0.5, color='red', markersize=25, edgecolor='k',  
 label='Local Authority (LA)')  
  
ax.axis('off')  
  
# an OpenStreetView basemap is added using contextily  
ctx.add\_basemap(ax)  
  
ax.legend(fontsize=12)  
  
plt.show()

|  |
| --- |
| Figure 2: Geographic location of LAs on a map of the UK |

We can see a few things that need to be considered, as the dataset is:

* Restricted to England LAs
* Incomplete for England, excluding key areas such as Cheltenham, Exeter and Oxford. This could introduce sample selection bias
* Heavily skewed towards the South East and London (although this may reflect real-world commuting patterns)

If we look take a look a random subsample of routes…

unique\_routes = LAcommute[['area\_name\_origin',  
 'area\_name\_dest']].drop\_duplicates()  
random\_routes = unique\_routes.sample(n=3, random\_state=0)  
  
plotting\_routes = LAcommute.merge(  
 random\_routes,  
 on=['area\_name\_origin', 'area\_name\_dest']  
)  
plotting\_routes['route'] = plotting\_routes['area\_name\_origin'] + \  
 ' to ' + plotting\_routes['area\_name\_dest']  
  
plt.figure(figsize=(12, 6))  
sns.lineplot(data=plotting\_routes, x='date', y='journey\_score', hue='route')  
  
plt.xlabel('Date')  
plt.ylabel('Journey Score')  
plt.title('Journey Score Over Time for Selected Routes')  
plt.grid(True)  
plt.legend(title='Route')  
plt.show()

|  |
| --- |
| Figure 3: Line graph of journey scores for random routes |

And with 'journey\_count\_decile':

unique\_routes = LAcommute[['area\_name\_origin',  
 'area\_name\_dest']].drop\_duplicates()  
random\_routes = unique\_routes.sample(n=3, random\_state=0)  
  
plotting\_routes = LAcommute.merge(  
 random\_routes,  
 on=['area\_name\_origin', 'area\_name\_dest']  
)  
plotting\_routes['route'] = plotting\_routes['area\_name\_origin'] + \  
 ' to ' + plotting\_routes['area\_name\_dest']  
  
plt.figure(figsize=(12, 6))  
sns.lineplot(data=plotting\_routes, x='date', y='journey\_count\_decile', hue='route')  
  
plt.xlabel('Date')  
plt.ylabel('Journey Count Decile')  
plt.title('Journey Count Decile Over Time for Selected Routes')  
plt.grid(True)  
plt.legend(title='Route', loc='upper right')  
plt.show()

|  |
| --- |
| Figure 4: Line graph of journey count decile for random routes |

A few important things can be deduced about the wider dataset:

* There is significant variation in the 'journey\_score,' i.e. som,e routes becoming both more and less popular over time   - There is less variation in the 'jouney\_count\_decile' variable, which is likely a result of the aggregation process   - This will be critical for inferential design/feature selection, as the provided dataset suggested independent variables are all “point in time” rather than time series data
* The routes are not constant over time, or there is data missing    - This is shown via “Brent” to “Central Bedfordshire”, which does not show any data in December 2019
* There are some irregularities, such as the “Brent” route flatlining, which could be a result of data collection/input errors

To get a bit more insight, we can look at the means:

means = LAcommute.groupby(['area\_name\_origin', 'area\_name\_dest'])['journey\_score'].mean()  
route\_means = round(means,3).reset\_index()  
route\_means.columns = ['area\_name\_origin', 'area\_name\_dest', 'mean\_journey\_score']  
  
route\_means

|  | area\_name\_origin | area\_name\_dest | mean\_journey\_score |
| --- | --- | --- | --- |
| 0 | Barking and Dagenham | Barking and Dagenham | 1.155 |
| 1 | Barking and Dagenham | Brent | 0.000 |
| 2 | Barking and Dagenham | Camden | 0.010 |
| 3 | Barking and Dagenham | Enfield | 0.016 |
| 4 | Barking and Dagenham | Greenwich | 0.030 |
| ... | ... | ... | ... |
| 1388 | Wolverhampton | Wolverhampton | 1.190 |
| 1389 | York | East Riding of Yorkshire | 0.148 |
| 1390 | York | Leeds | 0.161 |
| 1391 | York | Wakefield | 0.000 |
| 1392 | York | York | 1.200 |

This shows the dataset is not standardised across routes, i.e. not reflecting relative popularity over time of the same route

From here on, information may be relevant to the analysis, so the file will be saved and renamed “LAcommute\_clean.csv.”

Geographical data will be dropped as it will be added later, where relevant for feature additions rather than exploratory data analysis.

|  | date | area\_name\_origin | area\_code\_origin | area\_name\_dest | area\_code\_dest | journey\_score | journey\_count\_decile | distance | population\_origin | population\_dest | ... | travel\_time\_dest | gcse\_rate\_dest | life\_satisfaction\_dest | housing\_growth\_dest | lat\_origin | long\_origin | lat\_dest | long\_dest | geometry\_origin | geometry\_dest |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 2019-01-01 | Hartlepool | E06000001 | Hartlepool | E06000001 | 1.4414 | 9 | 0.000000 | 92401 | 92401 | ... | 12.9 | 67.6 | 7.33 | 161 | 54.676140 | -1.27018 | 54.676140 | -1.27018 | POINT (-1.27018 54.67614) | POINT (-1.27018 54.67614) |
| 1 | 2019-01-01 | Hartlepool | E06000001 | County Durham | E06000047 | -0.3129 | 3 | 37592.170378 | 92401 | 518562 | ... | 14.1 | 67.6 | 7.43 | 1343 | 54.676140 | -1.27018 | 54.685131 | -1.84050 | POINT (-1.27018 54.67614) | POINT (-1.84050 54.68513) |
| 2 | 2019-01-01 | Middlesbrough | E06000002 | Middlesbrough | E06000002 | 1.0253 | 10 | 0.000000 | 142134 | 142134 | ... | 15.4 | 63.2 | 7.21 | 456 | 54.544670 | -1.21099 | 54.544670 | -1.21099 | POINT (-1.21099 54.54467) | POINT (-1.21099 54.54467) |
| 3 | 2019-01-01 | Middlesbrough | E06000002 | Redcar and Cleveland | E06000003 | 0.3086 | 7 | 13069.176565 | 142134 | 136699 | ... | 13.3 | 69.6 | 7.44 | 365 | 54.544670 | -1.21099 | 54.567520 | -1.00608 | POINT (-1.21099 54.54467) | POINT (-1.00608 54.56752) |
| 4 | 2019-01-01 | Middlesbrough | E06000002 | Stockton-on-Tees | E06000004 | 0.3772 | 8 | 7379.212731 | 142134 | 196860 | ... | 13.2 | 69.5 | 7.40 | 616 | 54.544670 | -1.21099 | 54.556911 | -1.30664 | POINT (-1.21099 54.54467) | POINT (-1.30664 54.55691) |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 301022 | 2019-12-31 | Westminster | E09000033 | Sutton | E09000029 | -0.6732 | 4 | 16964.439602 | 208415 | 208516 | ... | 9.0 | 82.0 | 7.36 | 313 | 51.512199 | -0.15295 | 51.357559 | -0.17227 | POINT (-0.15295 51.51220) | POINT (-0.17227 51.35756) |
| 301023 | 2019-12-31 | Westminster | E09000033 | Tower Hamlets | E09000030 | 1.3720 | 10 | 8616.142460 | 208415 | 305066 | ... | 4.4 | 72.4 | 7.13 | 3248 | 51.512199 | -0.15295 | 51.515541 | -0.03643 | POINT (-0.15295 51.51220) | POINT (-0.03643 51.51554) |
| 301024 | 2019-12-31 | Westminster | E09000033 | Waltham Forest | E09000031 | -0.4483 | 6 | 13672.865893 | 208415 | 281015 | ... | 7.2 | 71.5 | 7.30 | 1263 | 51.512199 | -0.15295 | 51.594608 | -0.01881 | POINT (-0.15295 51.51220) | POINT (-0.01881 51.59461) |
| 301025 | 2019-12-31 | Westminster | E09000033 | Wandsworth | E09000032 | 0.1871 | 8 | 7117.584240 | 208415 | 334558 | ... | 6.2 | 74.2 | 7.34 | 1415 | 51.512199 | -0.15295 | 51.452400 | -0.20023 | POINT (-0.15295 51.51220) | POINT (-0.20023 51.45240) |
| 301026 | 2019-12-31 | Westminster | E09000033 | Westminster | E09000033 | 1.6534 | 10 | 0.000000 | 208415 | 208415 | ... | 5.1 | 77.3 | 7.21 | 580 | 51.512199 | -0.15295 | 51.512199 | -0.15295 | POINT (-0.15295 51.51220) | POINT (-0.15295 51.51220) |

LAcommute = LAcommute.drop(columns=[  
 'lat\_origin', 'long\_origin', 'lat\_dest', 'long\_dest', 'geometry\_origin', 'geometry\_dest'])  
  
LAcommute.to\_csv(  
 os.path.join(base\_data\_dir, "LAcommute\_clean.csv"), index=True  
)

## 6.3 Appendix C: Centrality

Exploring centrality using network graphs:

I will look at degree centrality for undirected networks, i.e. who has the most connections in terms of the number of links per node, although there are other measures of centrality.

|  | date | area\_name\_origin | area\_code\_origin | area\_name\_dest | area\_code\_dest | journey\_score | journey\_count\_decile | distance | population\_origin | population\_dest | ... | median\_weekly\_pay\_dest | emp\_rate\_dest | travel\_time\_dest | gcse\_rate\_dest | life\_satisfaction\_dest | housing\_growth\_dest | lat\_origin | long\_origin | lat\_dest | long\_dest |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 2019-01-02 | Hartlepool | E06000001 | Stockton-on-Tees | E06000004 | 1.5423 | 5 | 12951.008041 | 92401 | 196860 | ... | 469.4 | 74.8 | 13.2 | 69.5 | 7.40 | 616 | 54.676140 | -1.27018 | 54.556911 | -1.30664 |
| 1 | 2019-01-02 | Hartlepool | E06000001 | County Durham | E06000047 | 1.5683 | 5 | 37592.170378 | 92401 | 518562 | ... | 469.4 | 71.4 | 14.1 | 67.6 | 7.43 | 1343 | 54.676140 | -1.27018 | 54.685131 | -1.84050 |
| 2 | 2019-01-02 | Middlesbrough | E06000002 | Redcar and Cleveland | E06000003 | 0.9239 | 8 | 13069.176565 | 142134 | 136699 | ... | 439.2 | 68.4 | 13.3 | 69.6 | 7.44 | 365 | 54.544670 | -1.21099 | 54.567520 | -1.00608 |
| 3 | 2019-01-02 | Middlesbrough | E06000002 | Stockton-on-Tees | E06000004 | 1.8569 | 9 | 7379.212731 | 142134 | 196860 | ... | 469.4 | 74.8 | 13.2 | 69.5 | 7.40 | 616 | 54.544670 | -1.21099 | 54.556911 | -1.30664 |
| 4 | 2019-01-02 | Middlesbrough | E06000002 | County Durham | E06000047 | -1.0400 | 1 | 43441.543055 | 142134 | 518562 | ... | 469.4 | 71.4 | 14.1 | 67.6 | 7.43 | 1343 | 54.544670 | -1.21099 | 54.685131 | -1.84050 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 188229 | 2019-12-31 | Westminster | E09000033 | Southwark | E09000028 | 1.0626 | 10 | 7319.207505 | 208415 | 312591 | ... | 631.9 | 79.4 | 6.1 | 71.0 | 7.17 | 1096 | 51.512199 | -0.15295 | 51.465919 | -0.07306 |
| 188230 | 2019-12-31 | Westminster | E09000033 | Sutton | E09000029 | -0.6732 | 4 | 16964.439602 | 208415 | 208516 | ... | 565.8 | 77.4 | 9.0 | 82.0 | 7.36 | 313 | 51.512199 | -0.15295 | 51.357559 | -0.17227 |
| 188231 | 2019-12-31 | Westminster | E09000033 | Tower Hamlets | E09000030 | 1.3720 | 10 | 8616.142460 | 208415 | 305066 | ... | 680.3 | 74.4 | 4.4 | 72.4 | 7.13 | 3248 | 51.512199 | -0.15295 | 51.515541 | -0.03643 |
| 188232 | 2019-12-31 | Westminster | E09000033 | Waltham Forest | E09000031 | -0.4483 | 6 | 13672.865893 | 208415 | 281015 | ... | 624.7 | 71.5 | 7.2 | 71.5 | 7.30 | 1263 | 51.512199 | -0.15295 | 51.594608 | -0.01881 |
| 188233 | 2019-12-31 | Westminster | E09000033 | Wandsworth | E09000032 | 0.1871 | 8 | 7117.584240 | 208415 | 334558 | ... | 746.7 | 84.9 | 6.2 | 74.2 | 7.34 | 1415 | 51.512199 | -0.15295 | 51.452400 | -0.20023 |

G = nx.Graph()  
for \_, row in commute\_centr.iterrows():  
 G.add\_edge(row['area\_name\_origin'], row['area\_name\_dest'])

plt.figure(figsize=(24, 24))  
pos = nx.circular\_layout(G)  
nx.draw(  
 G, pos, with\_labels=False,  
 node\_size=500, node\_color='orange',  
 edge\_color='gray'  
)  
for node, (x, y) in pos.items():  
 angle = np.arctan2(y, x)  
 rotation = np.degrees(angle)  
 if x < 0:  
 rotation += 180  
 plt.text(  
 x, y, node,  
 fontsize=13, color='black',  
 ha='center', va='center',  
 rotation=rotation, rotation\_mode='anchor'  
 )  
plt.title('Network Graph of Commuting Routes', fontsize=28)  
plt.show()

|  |
| --- |
| Figure 5: Network Graph of Commuting Routes |

## 6.4 Appendix D: Data Diagnostics

commute\_trans

|  | pairs | journey\_score | journey\_count\_decile | distance | |\_population\_diff\_| | |\_value\_added\_hourly\_diff\_| | |\_median\_weekly\_pay\_diff\_| | |\_emp\_rate\_diff\_| | |\_travel\_time\_diff\_| | |\_gcse\_rate\_diff\_| | |\_life\_satisfaction\_diff\_| | |\_housing\_growth\_diff\_| | |\_avg\_monthly\_rent\_diff\_| | |\_centrality\_diff\_| | route\_midpoint\_(geo) |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | Barking and Dagenham - Barnet | -0.010609 | 1.363636 | 25056.274464 | 171762 | 0.09 | 51.4 | 8.3 | 0.9 | 13.4 | 0.05 | 1202 | 150.0 | 0.017241 | POINT (-0.04437 51.57832) |
| 1 | Barking and Dagenham - Brent | -0.397323 | 1.291667 | 27881.522854 | 128596 | 0.07 | 29.6 | 3.1 | 0.9 | 3.5 | 0.10 | 1356 | 252.0 | 0.068966 | POINT (-0.07310 51.55498) |
| 2 | Barking and Dagenham - Camden | 0.112844 | 2.959443 | 20345.000863 | 1692 | 14.43 | 170.7 | 2.3 | 2.7 | 4.3 | 0.57 | 539 | 858.0 | 0.146552 | POINT (-0.01671 51.54431) |
| 3 | Barking and Dagenham - Enfield | -0.004368 | 2.500977 | 19482.245346 | 116323 | 5.36 | 15.6 | 2.5 | 0.1 | 0.7 | 0.49 | 251 | 50.0 | 0.000000 | POINT (0.02400 51.59722) |
| 4 | Barking and Dagenham - Greenwich | 0.114385 | 2.973163 | 9642.046411 | 69377 | 1.56 | 95.4 | 8.3 | 0.0 | 0.8 | 0.13 | 6 | 150.0 | 0.051724 | POINT (0.08979 51.50474) |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 672 | West Berkshire - Wokingham | 0.163863 | 2.844894 | 29259.923429 | 10906 | 4.18 | 64.6 | 5.8 | 1.6 | 5.6 | 0.01 | 512 | 175.0 | 0.008621 | POINT (-1.08649 51.43427) |
| 673 | Westminster - Wiltshire | 0.075996 | 1.931034 | 126401.984988 | 294570 | 21.88 | 290.4 | 10.8 | 9.5 | 4.9 | 0.29 | 2266 | 1633.0 | 0.344828 | POINT (-1.03978 51.42052) |
| 674 | Westminster - Windsor and Maidenhead | 0.026557 | 2.449654 | 37615.785116 | 54522 | 0.44 | 164.8 | 11.9 | 4.6 | 0.8 | 0.29 | 270 | 1183.0 | 0.318966 | POINT (-0.41418 51.49627) |
| 675 | Westminster - Wokingham | -0.168467 | 1.416667 | 51144.513300 | 36626 | 4.76 | 113.0 | 10.0 | 6.8 | 3.1 | 0.12 | 549 | 1283.0 | 0.344828 | POINT (-0.52615 51.46758) |
| 676 | Windsor and Maidenhead - Wokingham | 0.161748 | 2.881737 | 14209.316238 | 17896 | 5.20 | 51.8 | 1.9 | 2.2 | 2.3 | 0.17 | 819 | 100.0 | 0.025862 | POINT (-0.78738 51.45165) |

To assess potential concerns in the data that might require transformations, we can use histogram visualisations, scatter plots and statistical tests:

numeric\_cols = commute\_trans.select\_dtypes(  
 include=['float64', 'int64']).columns  
n\_cols = 3  
n\_rows = (len(numeric\_cols) + n\_cols - 1) // n\_cols  
  
fig, axes = plt.subplots(n\_rows, n\_cols, figsize=(15, n\_rows \* 5))  
axes = axes.flatten()  
  
palette = ['#69b3a2']  
sns.set\_theme(style='darkgrid')  
  
for i, col in enumerate(numeric\_cols):  
 sns.histplot(commute\_trans[col], kde=True,  
 color=palette[0], bins=30, ax=axes[i])  
 axes[i].set\_title(f'Distribution of {col}')  
 axes[i].set\_xlabel(col)  
 axes[i].set\_ylabel('Frequency')  
  
for j in range(len(numeric\_cols), len(axes)):  
 fig.delaxes(axes[j])  
  
plt.tight\_layout()  
plt.show()

|  |
| --- |
| Figure 6: Histogram of distribution of commute data |

Almost all variables are rightly skewed which could be a concern for OLS, as it violates assumptions of normality.

fig, axes = plt.subplots(n\_rows, n\_cols, figsize=(15, n\_rows \* 5))  
axes = axes.flatten()  
  
palette = ['#ffb347']  
sns.set\_theme(style='darkgrid')  
  
for i, col in enumerate(numeric\_cols):  
 sns.boxplot(y=commute\_trans[col], color=palette[0], ax=axes[i])  
 axes[i].set\_title(f'Box Plot of {col}')  
 axes[i].set\_xlabel('')  
 axes[i].set\_ylabel(col)  
  
for j in range(len(numeric\_cols), len(axes)):  
 fig.delaxes(axes[j])  
  
plt.tight\_layout()  
plt.show()

|  |
| --- |
| Figure 7: Box plot of distribution of commute data |

In addition we can look at the relationship between 'journey\_score' and the other variables to assess linearity. This is important for OLS, as it assumes a linear relationship between the dependent and independent variables:

y\_sp = 'journey\_score'  
X\_sp = [col for col in numeric\_cols if col != y\_sp]  
  
fig, axes = plt.subplots(n\_rows, n\_cols, figsize=(15, n\_rows \* 5))  
axes = axes.flatten()  
  
palette = ['#ff6961']  
sns.set\_theme(style='darkgrid')  
  
for i, col in enumerate(X\_sp):  
 sns.scatterplot(  
 x=commute\_trans[col], y=commute\_trans[y\_sp], color=palette[0], ax=axes[i])  
 axes[i].set\_title(f'{col} vs {y\_sp}')  
 axes[i].set\_xlabel(col)  
 axes[i].set\_ylabel(y\_sp)  
  
for j in range(len(X\_sp), len(axes)):  
 fig.delaxes(axes[j])  
  
plt.tight\_layout()  
plt.show()

|  |
| --- |
| Figure 8: Scatterplot of distribution of commute data |

… the concerns around non-linearity are confirmed by the scatter plot all plots show show variance of 'journey\_score' increasing with X. This therefore violates a key OLS assumption.