exercise2

March 18, 2025

[1]:
```
import pandas as pd

# Read data from xlsx file
df_trips = pd.read_excel('PFI_2025_ex2_data.xlsx')

df_trips
```

```
/home/mbg/.local/lib/python3.10/site-packages/openpyxl/styles/stylesheet.py:237:
UserWarning: Workbook contains no default style, apply openpyxl's default
  warn("Workbook contains no default style, apply openpyxl's default")
```

[1]:

	ResiZone	DestZone	PopResi	EmpResi	PopDest	EmpDest	CarStat	Dist	ae	cc	ct	pc	pt
0	1	1	15446.270280	8990.436751	15446.270280	8990.436751	0.90	0.289223	6.922697	0.242947	0.216917	15.0	
1	1	2	15446.270280	8990.436751	8431.287835	5653.883233	0.90	2.612138	7.759704	2.194196	1.959104	15.0	
2	1	3	15446.270280	8990.436751	13526.411712	9921.381329	0.90	4.929963	7.397353	4.141169	3.697473	15.0	
3	1	4	15446.270280	8990.436751	8663.696994	5979.914469	0.90	7.037586	7.387384	5.911573	5.278190	15.0	
4	1	5	15446.270280	8990.436751	14782.811654	12480.475745	0.90	9.314707	7.908371	7.824354	6.986031	15.0	
..	…	…	…	…	…	…	…	…	…	…	…	…	
395	20	16	16808.942787	17720.048513	12576.911044	7323.163165	0.99	9.550852	7.605019	8.022716	7.163139	15.0	
396	20	17	16808.942787	17720.048513	5608.609071	4747.014808	0.99	7.420676	7.840353	6.233368	5.565507	15.0	
397	20	18	16808.942787	17720.048513	1403.393770	9993.609788	0.99	4.944903	7.851580	4.153719	3.708677	15.0	
398	20	19	16808.942787	17720.048513	12938.436403	16131.022424	0.99	2.798100	8.199186	2.350404	2.098575	15.0	
399	20	20	16808.942787	17720.048513	16808.942787	17720.048513	0.99	0.925344	7.848028	0.777289	0.694008	15.0	

```
0    0.247905
1    2.238976
2    4.225683
3    6.032217
4    7.984035
..        …
395  8.186444
396  6.360579
397  4.238488
398  2.398372
399  0.793152

[400 rows x 13 columns]
```

[2]:
```
df_trips.describe()
```

[2]:

	ResiZone	DestZone	PopResi	EmpResi	PopDest
count	400.000000	400.000000	400.000000	400.000000	400.000000
mean	10.500000	10.500000	11448.634100	10315.578174	11448.634100
std	5.773503	5.773503	5329.997306	4526.945887	5329.997306
min	1.000000	1.000000	1326.917251	4568.450007	1326.917251
25%	5.750000	5.750000	8007.687380	6447.142311	8007.687380
50%	10.500000	10.500000	11734.179541	9344.055247	11734.179541
75%	15.250000	15.250000	15498.165060	13393.112415	15498.165060
max	20.000000	20.000000	19842.180276	19400.363349	19842.180276

	EmpDest	CarStat	Dist	ae	cc
count	400.000000	400.000000	400.000000	400.000000	400.000000
mean	10315.578174	0.899000	15.111967	7.512799	12.694052
std	4526.945887	0.060978	10.405811	0.420873	8.740881
min	4568.450007	0.760000	0.028840	6.219225	0.024226
25%	6447.142311	0.850000	6.961989	7.229553	5.848071
50%	9344.055247	0.905000	13.194085	7.501645	11.083032
75%	13393.112415	0.932500	22.368325	7.811129	18.789393
max	19400.363349	1.040000	45.059042	8.820077	37.849595

	ct	pc	pt
count	400.000000	400.000000	400.000000
mean	11.333975	19.406786	12.953115
std	7.804358	6.767564	8.919266
min	0.021630	15.000000	0.024720
25%	5.221492	15.000000	5.967419
50%	9.895564	15.000000	11.309216
75%	16.776244	22.368325	19.172850
max	33.794281	45.059042	38.622036

[3]:
```
df_od = pd.read_excel('PFI_2025_ex2_od.xlsx')

df_od
```

```
/home/mbg/.local/lib/python3.10/site-packages/openpyxl/styles/stylesheet.py:237:
UserWarning: Workbook contains no default style, apply openpyxl's default
  warn("Workbook contains no default style, apply openpyxl's default")
```

[3]:
```
     FromID  ToID     trip_w       trip_b       trip_c     trip_cp  \
0         1     1  987.721585  2041.582022  1215.402571  203.684045
1         1     2   76.442781   556.483145   998.942044  145.924486
2         1     3    8.540904   197.942473   965.779160  178.100343
3         1     4    1.208151    57.897811   803.238484  112.660213
4         1     5    0.110915    31.357639   923.052825  151.156707
..      …   …         …           …            …          …
395      20    16    0.131775    14.531779   617.614480   90.500037
396      20    17    0.997029    49.967352   584.830000  119.511398
397      20    18    7.688951   174.598898   816.228705  135.064878
398      20    19   84.959797   473.586815  1100.833369  172.875274
399      20    20  385.026680  1188.602110  1300.928962  181.479500

         trip_p
0    141.907615
1    135.584200
2    166.219377
3    134.129543
4    146.088133
..          …
395  106.698972
396  101.353441
397  123.562731
398  154.722094
399  120.224187

[400 rows x 7 columns]
```

[4]:
```
df_od.describe()
```

[4]:
```
           FromID        ToID        trip_w       trip_b       trip_c  \
count  400.000000  400.000000  4.000000e+02   400.000000   400.000000
mean    10.500000   10.500000  3.326124e+01   117.197367   514.583251
std      5.773503    5.773503  1.215471e+02   275.625013   278.154546
min      1.000000    1.000000  4.894137e-13     0.000141    95.758718
25%      5.750000    5.750000  1.568114e-05     0.082061   282.653554
50%     10.500000   10.500000  9.389495e-03     2.707815   482.223772
75%     15.250000   15.250000  1.220511e+00    62.514887   689.911731
max     20.000000   20.000000  9.877216e+02  2041.582022  1387.817219
```

```
          trip_cp      trip_p
count  400.000000  400.000000
mean    72.902472   87.741329
std     49.423098   37.676060
min      8.350678   20.045448
25%     30.976264   57.288568
50%     60.430154   88.905113
75%    106.644124  115.101918
max    258.077742  237.857125
```

[5]:
```
params = {
    'k_walk': 1.5, # Walk constant
    'k_bike': 2, # Bike constant
    'k_car': 0.5, # Car constant
    'k_carp': -0.5, # Carpool constant
    'beta_wt': -0.12, # Walk time parameter (U/min)
    'beta_bt': -0.12, # Bike time parameter (U/min)
    'beta_cc': -.05, # Car cost parameter (U/DKK)
    'beta_ct': -0.06, # Car time parameter (U/min)
    'beta_cstat': 1, # Car status parameter
    'beta_cpt': -.1, # Car passenger time parameter (U/min)
    'beta_pc': -0.05, # Public transport cost parameter (U/DKK)
    'beta_pt': -0.05, # Public transport time parameter (U/min)
    'beta_ae': -0.03, # Public transport access/egress parameter (U/min)
    'mu': 0.7, # Logsum parameter
    'alpha': 1, # Size parameter
}

WALKING_SPEED = 6 # km/h
BIKING_SPEED = 12 # km/h
districts = [i for i in range(1, 21)]

# Calculate walking time on all districts
```

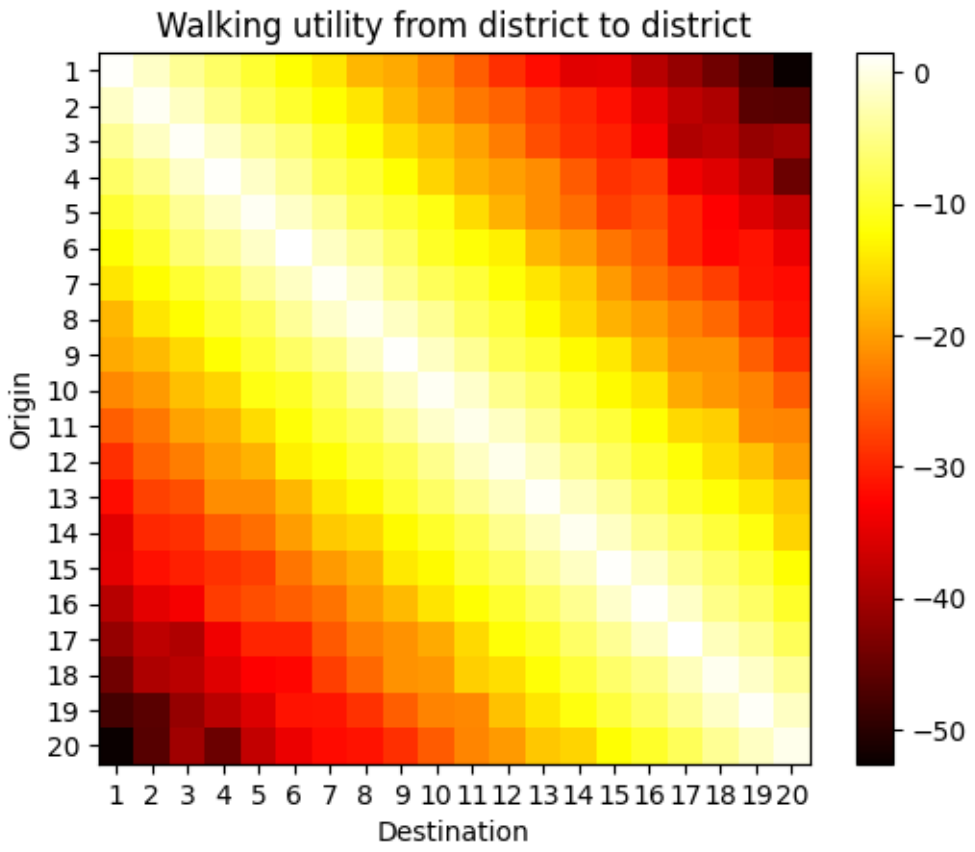
[6]:
```
import numpy as np

# Calculate the walking utility functions for all districts
# V_n(walk|district) = k_walk + beta_wt * 60 * distance/walking_speed

def utility_walk(d_from, d_to, alpha=params['k_walk']):
    # Find the on row in df_zones that have
    # ResiZone = d_from
    # DestZone = d_to
    row = df_trips[(df_trips['ResiZone'] == d_from) & (df_trips['DestZone'] ==␣
 ↪d_to)]
    distance = row['Dist'].values[0]
```

```
    return alpha + params['beta_wt'] * 60 * distance/WALKING_SPEED

def utility_bike(d_from, d_to, alpha=params['k_bike'], df_trips=df_trips):
    # Find the on row in df_zones that have
    # ResiZone = d_from
    # DestZone = d_to
    row = df_trips[(df_trips['ResiZone'] == d_from) & (df_trips['DestZone'] ==␣
 ↪d_to)]
    distance = row['Dist'].values[0]
    return alpha + params['beta_bt'] * 60 * distance/BIKING_SPEED

def utility_car(d_from, d_to, alpha=params['k_car']):
    # Find the on row in df_zones that have
    # ResiZone = d_from
    # DestZone = d_to
    # V_n(car|district) = k_car + beta_cc * cc(district) + beta_ct *␣
 ↪ct(district) + beta_cstat * CarStat[district]
    district = df_trips[(df_trips['ResiZone'] == d_from) &␣
 ↪(df_trips['DestZone'] == d_to)]
    cc = district['cc'].values[0]
    ct = district['ct'].values[0]
    car_stat = district['CarStat'].values[0]
    return alpha + params['beta_cc'] * cc + params['beta_ct'] * ct +␣
 ↪params['beta_cstat'] * car_stat

def utility_carpool(d_from, d_to, alpha=params['k_carp']):
    # Find the on row in df_zones that have
    # ResiZone = d_from
    # DestZone = d_to
    # V_n(carpool|district) = k_carp + beta_carp * ct(district)
    district = df_trips[(df_trips['ResiZone'] == d_from) &␣
 ↪(df_trips['DestZone'] == d_to)]
    ct = district['ct'].values[0]
    return alpha + params['beta_cpt'] * ct

def utility_public_transport(d_from, d_to):
    # Find the on row in df_zones that have
    # ResiZone = d_from
    # DestZone = d_to
    # V_n(pub|district) = beta_pc * pc(district) + beta_pt * pt(district) +␣
 ↪beta_ae * ae(district)
    district = df_trips[(df_trips['ResiZone'] == d_from) &␣
 ↪(df_trips['DestZone'] == d_to)]
    pc = district['pc'].values[0]
    pt = district['pt'].values[0]
    ae = district['ae'].values[0]
```

```
    return params['beta_pc'] * pc + params['beta_pt'] * pt + params['beta_ae']␣
 ↪* ae

def destination_utility(d_from, d_to):
    # Calculate the utility for a destination given a starting point
    # V_n(destination) = alpha * ln(Emp(destination) + 0.15 * Pop(destination))
    # Emp(destination) = Employment in destination, EmpDest
    # Pop(destination) = Population in destination, PopDest
    route = df_trips[(df_trips['ResiZone'] == d_from) & (df_trips['DestZone']␣
 ↪== d_to)]
    emp_dest = route['EmpDest'].values[0]
    pop_dest = route['PopDest'].values[0]
    return params['alpha'] * np.log(emp_dest + 0.15 * pop_dest)
```

[7]:
```
import matplotlib.pyplot as plt

# Pretty print utility to and from all districts by walk
# Whilst also plotting a heatmap of the destinations

walking_matrix = pd.DataFrame(index=districts, columns=districts)

for d_from in districts:
    for d_to in districts:
        walking_matrix.at[d_from, d_to] = utility_walk(d_from, d_to)

# Convert to plottable format
walking_matrix = walking_matrix.astype(float)

plt.imshow(walking_matrix, cmap='hot', interpolation='nearest')
plt.colorbar()
# Make labels on each column in the middle as ints
plt.xticks(np.arange(0, len(districts), 1), districts)
plt.yticks(np.arange(0, len(districts), 1), districts)
plt.xlabel('Destination')
plt.ylabel('Origin')
plt.title('Walking utility from district to district')
plt.show()

walking_matrix.round(1)
```

```
/home/mbg/.local/lib/python3.10/site-
packages/matplotlib/projections/__init__.py:63: UserWarning: Unable to import
Axes3D. This may be due to multiple versions of Matplotlib being installed (e.g.
as a system package and as a pip package). As a result, the 3D projection is not
available.
  warnings.warn("Unable to import Axes3D. This may be due to multiple versions
of "
```



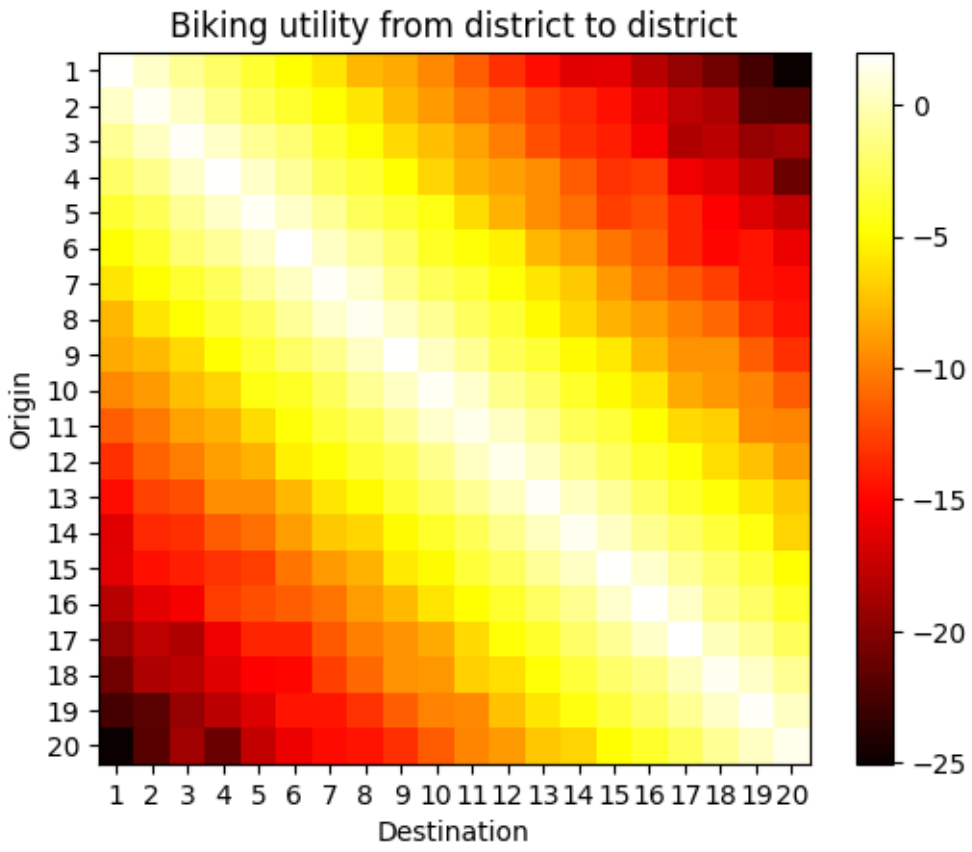
[7]:

	1	2	3	4	5	6	7	8	9	10	11	12	\
1	1.2	-1.6	-4.4	-6.9	-9.7	-12.0	-14.3	-18.1	-19.1	-21.9	-25.3	-29.0	
2	-1.6	0.7	-1.7	-4.9	-7.7	-9.7	-12.3	-14.3	-17.8	-20.4	-23.2	-24.8	
3	-4.4	-1.7	1.0	-1.6	-4.4	-6.2	-9.6	-12.3	-15.3	-17.4	-19.8	-22.6	
4	-6.9	-4.9	-1.6	1.2	-1.6	-4.0	-7.5	-9.3	-12.2	-15.6	-18.4	-19.9	
5	-9.7	-7.7	-4.4	-1.6	0.7	-1.6	-4.2	-7.5	-9.3	-11.4	-15.1	-18.5	
6	-12.0	-9.7	-6.2	-4.0	-1.6	1.3	-1.9	-4.2	-7.0	-10.2	-12.0	-13.4	
7	-14.3	-12.3	-9.6	-7.5	-4.2	-1.9	0.9	-1.5	-4.7	-7.6	-9.3	-11.9	
8	-18.1	-14.3	-12.3	-9.3	-7.5	-4.2	-1.5	0.5	-1.8	-4.3	-7.3	-9.5	
9	-19.1	-17.8	-15.3	-12.2	-9.3	-7.0	-4.7	-1.8	1.1	-1.8	-4.4	-7.8	
10	-21.9	-20.4	-17.4	-15.6	-11.4	-10.2	-7.6	-4.3	-1.8	0.7	-1.3	-4.8	
11	-25.3	-23.2	-19.8	-18.4	-15.1	-12.0	-9.3	-7.3	-4.4	-1.3	0.3	-1.8	
12	-29.0	-24.8	-22.6	-19.9	-18.5	-13.4	-11.9	-9.5	-7.8	-4.8	-1.8	0.3	
13	-31.8	-27.6	-26.6	-21.3	-21.4	-18.1	-14.3	-12.5	-9.5	-6.9	-4.4	-2.1	
14	-35.2	-29.7	-29.0	-25.4	-24.0	-20.2	-16.6	-15.5	-12.6	-10.1	-7.7	-4.7	
15	-35.0	-31.5	-30.4	-28.8	-27.8	-23.4	-20.4	-18.4	-14.1	-12.5	-9.3	-7.3	
16	-38.5	-34.9	-33.4	-27.9	-26.5	-25.2	-23.6	-20.3	-17.9	-14.4	-12.1	-9.9	
17	-41.2	-38.2	-39.1	-33.9	-29.8	-29.9	-25.7	-22.5	-21.0	-19.1	-15.3	-12.0	
18	-44.3	-39.5	-38.3	-35.3	-32.9	-32.3	-27.7	-24.3	-21.0	-20.6	-16.2	-14.8	
19	-47.7	-46.1	-41.3	-38.3	-35.5	-31.3	-31.1	-28.7	-25.2	-22.2	-21.8	-17.2	

```
20 -52.6 -46.3 -40.5 -44.7 -37.7 -34.2 -31.9 -31.3 -29.0 -25.4 -22.1 -20.5
```

```
      13    14    15    16    17    18    19    20
1  -31.8 -35.2 -35.0 -38.5 -41.2 -44.3 -47.7 -52.6
2  -27.6 -29.7 -31.5 -34.9 -38.2 -39.5 -46.1 -46.3
3  -26.6 -29.0 -30.4 -33.4 -39.1 -38.3 -41.3 -40.5
4  -21.3 -25.4 -28.8 -27.9 -33.9 -35.3 -38.3 -44.7
5  -21.4 -24.0 -27.8 -26.5 -29.8 -32.9 -35.5 -37.7
6  -18.1 -20.2 -23.4 -25.2 -29.9 -32.3 -31.3 -34.2
7  -14.3 -16.6 -20.4 -23.6 -25.7 -27.7 -31.1 -31.9
8  -12.5 -15.5 -18.4 -20.3 -22.5 -24.3 -28.7 -31.3
9   -9.5 -12.6 -14.1 -17.9 -21.0 -21.0 -25.2 -29.0
10  -6.9 -10.1 -12.5 -14.4 -19.1 -20.6 -22.2 -25.4
11  -4.4  -7.7  -9.3 -12.1 -15.3 -16.2 -21.8 -22.1
12  -2.1  -4.7  -7.3  -9.9 -12.0 -14.8 -17.2 -20.5
13   0.9  -1.9  -4.2  -7.1 -10.0 -11.9 -14.2 -16.7
14  -1.9   0.6  -1.7  -4.6  -6.8  -9.3 -11.6 -15.8
15  -4.2  -1.7   1.1  -1.3  -4.3  -6.6  -9.0 -12.2
16  -7.1  -4.6  -1.3   1.2  -1.6  -4.9  -6.9 -10.0
17 -10.0  -6.8  -4.3  -1.6   1.5  -2.1  -4.4  -7.4
18 -11.9  -9.3  -6.6  -4.9  -2.1   0.5  -1.6  -4.4
19 -14.2 -11.6  -9.0  -6.9  -4.4  -1.6   0.8  -1.9
20 -16.7 -15.8 -12.2 -10.0  -7.4  -4.4  -1.9   0.4
```

[8]:
```python
# Repeat for biking
biking_matrix = pd.DataFrame(index=districts, columns=districts)

for d_from in districts:
    for d_to in districts:
        biking_matrix.at[d_from, d_to] = utility_bike(d_from, d_to)

# Convert to plottable format
biking_matrix = biking_matrix.astype(float)

plt.imshow(biking_matrix, cmap='hot', interpolation='nearest')
plt.colorbar()
# Make labels on each column in the middle as ints
plt.xticks(np.arange(0, len(districts), 1), districts)
plt.yticks(np.arange(0, len(districts), 1), districts)
plt.xlabel('Destination')
plt.ylabel('Origin')
plt.title('Biking utility from district to district')
plt.show()

biking_matrix.round(1)
```



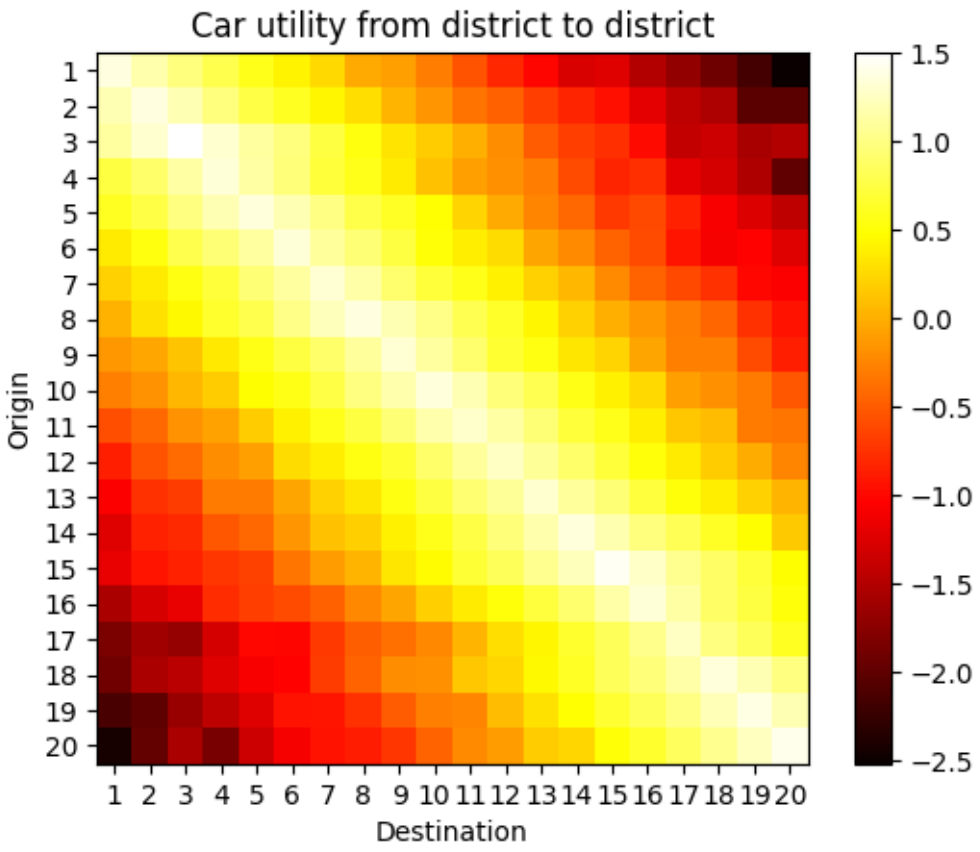
[8]:

	1	2	3	4	5	6	7	8	9	10	11	12	\
1	1.8	0.4	-1.0	-2.2	-3.6	-4.8	-5.9	-7.8	-8.3	-9.7	-11.4	-13.2	
2	0.4	1.6	0.4	-1.2	-2.6	-3.6	-4.9	-5.9	-7.7	-8.9	-10.3	-11.1	
3	-1.0	0.4	1.7	0.4	-0.9	-1.9	-3.6	-4.9	-6.4	-7.4	-8.6	-10.1	
4	-2.2	-1.2	0.4	1.8	0.4	-0.8	-2.5	-3.4	-4.9	-6.6	-8.0	-8.7	
5	-3.6	-2.6	-0.9	0.4	1.6	0.4	-0.9	-2.5	-3.4	-4.4	-6.3	-8.0	
6	-4.8	-3.6	-1.9	-0.8	0.4	1.9	0.3	-0.8	-2.2	-3.9	-4.7	-5.4	
7	-5.9	-4.9	-3.6	-2.5	-0.9	0.3	1.7	0.5	-1.1	-2.5	-3.4	-4.7	
8	-7.8	-5.9	-4.9	-3.4	-2.5	-0.8	0.5	1.5	0.3	-0.9	-2.4	-3.5	
9	-8.3	-7.7	-6.4	-4.9	-3.4	-2.2	-1.1	0.3	1.8	0.3	-1.0	-2.6	
10	-9.7	-8.9	-7.4	-6.6	-4.4	-3.9	-2.5	-0.9	0.3	1.6	0.6	-1.1	
11	-11.4	-10.3	-8.6	-8.0	-6.3	-4.7	-3.4	-2.4	-1.0	0.6	1.4	0.3	
12	-13.2	-11.1	-10.1	-8.7	-8.0	-5.4	-4.7	-3.5	-2.6	-1.1	0.3	1.4	
13	-14.7	-12.5	-12.0	-9.4	-9.5	-7.8	-5.9	-5.0	-3.5	-2.2	-0.9	0.2	
14	-16.4	-13.6	-13.3	-11.4	-10.7	-8.8	-7.0	-6.5	-5.1	-3.8	-2.6	-1.1	
15	-16.2	-14.5	-13.9	-13.1	-12.7	-10.5	-8.9	-7.9	-5.8	-5.0	-3.4	-2.4	
16	-18.0	-16.2	-15.4	-12.7	-12.0	-11.4	-10.5	-8.9	-7.7	-6.0	-4.8	-3.7	
17	-19.4	-17.9	-18.3	-15.7	-13.6	-13.7	-11.6	-10.0	-9.2	-8.3	-6.4	-4.8	
18	-20.9	-18.5	-17.9	-16.4	-15.2	-14.9	-12.6	-10.9	-9.2	-9.1	-6.9	-6.1	
19	-22.6	-21.8	-19.4	-17.9	-16.5	-14.4	-14.3	-13.1	-11.4	-9.9	-9.6	-7.4	

```
20 -25.0 -21.9 -19.0 -21.1 -17.6 -15.9 -14.7 -14.4 -13.3 -11.4  -9.8  -9.0
```

	13	14	15	16	17	18	19	20
1	-14.7	-16.4	-16.2	-18.0	-19.4	-20.9	-22.6	-25.0
2	-12.5	-13.6	-14.5	-16.2	-17.9	-18.5	-21.8	-21.9
3	-12.0	-13.3	-13.9	-15.4	-18.3	-17.9	-19.4	-19.0
4	-9.4	-11.4	-13.1	-12.7	-15.7	-16.4	-17.9	-21.1
5	-9.5	-10.7	-12.7	-12.0	-13.6	-15.2	-16.5	-17.6
6	-7.8	-8.8	-10.5	-11.4	-13.7	-14.9	-14.4	-15.9
7	-5.9	-7.0	-8.9	-10.5	-11.6	-12.6	-14.3	-14.7
8	-5.0	-6.5	-7.9	-8.9	-10.0	-10.9	-13.1	-14.4
9	-3.5	-5.1	-5.8	-7.7	-9.2	-9.2	-11.4	-13.3
10	-2.2	-3.8	-5.0	-6.0	-8.3	-9.1	-9.9	-11.4
11	-0.9	-2.6	-3.4	-4.8	-6.4	-6.9	-9.6	-9.8
12	0.2	-1.1	-2.4	-3.7	-4.8	-6.1	-7.4	-9.0
13	1.7	0.3	-0.8	-2.3	-3.7	-4.7	-5.9	-7.1
14	0.3	1.5	0.4	-1.0	-2.1	-3.4	-4.5	-6.6
15	-0.8	0.4	1.8	0.6	-0.9	-2.0	-3.3	-4.8
16	-2.3	-1.0	0.6	1.9	0.5	-1.2	-2.2	-3.7
17	-3.7	-2.1	-0.9	0.5	2.0	0.2	-0.9	-2.5
18	-4.7	-3.4	-2.0	-1.2	0.2	1.5	0.5	-1.0
19	-5.9	-4.5	-3.3	-2.2	-0.9	0.5	1.7	0.3
20	-7.1	-6.6	-4.8	-3.7	-2.5	-1.0	0.3	1.4

[9]:
```python
# Repeat for car
car_matrix = pd.DataFrame(index=districts, columns=districts)

for d_from in districts:
    for d_to in districts:
        car_matrix.at[d_from, d_to] = utility_car(d_from, d_to)

# Convert to plottable format
car_matrix = car_matrix.astype(float)

plt.imshow(car_matrix, cmap='hot', interpolation='nearest')
plt.colorbar()
# Make labels on each column in the middle as ints
plt.xticks(np.arange(0, len(districts), 1), districts)
plt.yticks(np.arange(0, len(districts), 1), districts)
plt.xlabel('Destination')
plt.ylabel('Origin')
plt.title('Car utility from district to district')
plt.show()

car_matrix.round(1)
```



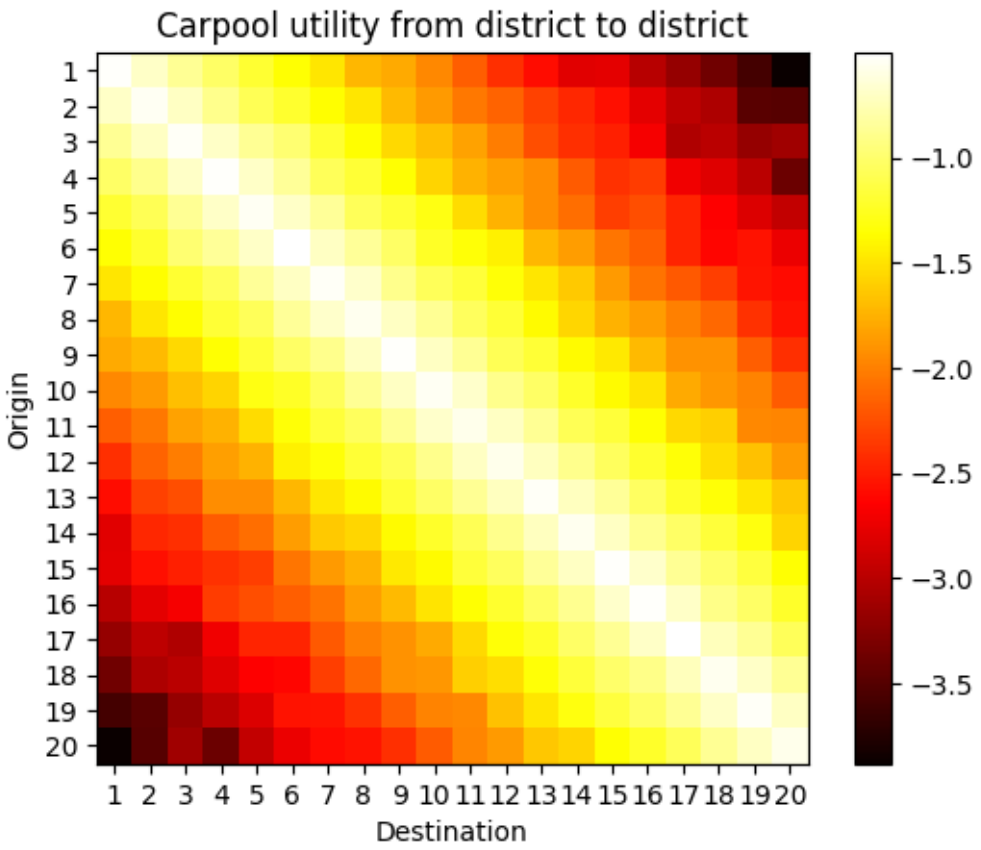
[9]:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	\
1	1.4	1.2	1.0	0.8	0.6	0.4	0.3	-0.0	-0.1	-0.3	-0.5	-0.8	-1.0	-1.3	-1.2	
2	1.2	1.4	1.2	1.0	0.8	0.6	0.4	0.3	0.0	-0.2	-0.4	-0.5	-0.7	-0.8	-1.0	
3	1.1	1.3	1.5	1.3	1.1	1.0	0.7	0.5	0.3	0.2	-0.0	-0.2	-0.5	-0.7	-0.8	
4	0.7	0.9	1.1	1.3	1.1	1.0	0.7	0.6	0.4	0.1	-0.1	-0.2	-0.3	-0.6	-0.8	
5	0.6	0.8	1.0	1.2	1.4	1.2	1.0	0.8	0.6	0.5	0.2	-0.0	-0.2	-0.4	-0.7	
6	0.4	0.5	0.8	0.9	1.1	1.3	1.1	0.9	0.7	0.5	0.4	0.3	-0.1	-0.2	-0.5	
7	0.2	0.4	0.6	0.7	0.9	1.1	1.3	1.1	0.9	0.7	0.6	0.4	0.2	0.1	-0.2	
8	0.0	0.3	0.4	0.7	0.8	1.0	1.2	1.4	1.2	1.0	0.8	0.6	0.4	0.2	-0.0	
9	-0.1	-0.0	0.1	0.4	0.6	0.7	0.9	1.1	1.3	1.1	0.9	0.7	0.6	0.3	0.2	
10	-0.3	-0.2	0.0	0.2	0.5	0.6	0.8	1.0	1.2	1.4	1.2	1.0	0.8	0.6	0.4	
11	-0.6	-0.4	-0.2	-0.1	0.2	0.4	0.6	0.7	0.9	1.2	1.3	1.1	0.9	0.7	0.6	
12	-0.9	-0.6	-0.4	-0.2	-0.1	0.3	0.4	0.6	0.7	0.9	1.1	1.3	1.1	0.9	0.7	
13	-1.1	-0.8	-0.7	-0.3	-0.3	-0.1	0.2	0.3	0.6	0.7	0.9	1.1	1.3	1.1	0.9	
14	-1.2	-0.8	-0.8	-0.5	-0.4	-0.2	0.1	0.2	0.4	0.6	0.8	1.0	1.2	1.4	1.2	
15	-1.2	-0.9	-0.8	-0.7	-0.7	-0.3	-0.1	0.0	0.3	0.5	0.7	0.8	1.1	1.2	1.4	
16	-1.6	-1.3	-1.2	-0.8	-0.7	-0.6	-0.5	-0.2	-0.1	0.2	0.4	0.5	0.7	0.9	1.1	
17	-1.8	-1.6	-1.7	-1.3	-1.0	-1.0	-0.7	-0.5	-0.4	-0.2	0.0	0.3	0.4	0.7	0.8	
18	-1.9	-1.6	-1.5	-1.2	-1.1	-1.0	-0.7	-0.5	-0.2	-0.2	0.1	0.2	0.4	0.6	0.8	
19	-2.1	-2.0	-1.7	-1.4	-1.2	-0.9	-0.9	-0.8	-0.5	-0.3	-0.2	0.1	0.3	0.5	0.7	

```
20 -2.4 -2.0 -1.6 -1.9 -1.4 -1.1 -0.9 -0.9 -0.7 -0.5 -0.2 -0.1  0.2  0.2  0.5
```

	16	17	18	19	20
1	-1.5	-1.7	-1.9	-2.2	-2.5
2	-1.2	-1.4	-1.5	-2.0	-2.0
3	-1.0	-1.4	-1.3	-1.6	-1.5
4	-0.8	-1.2	-1.3	-1.5	-2.0
5	-0.6	-0.8	-1.1	-1.3	-1.4
6	-0.6	-0.9	-1.1	-1.0	-1.2
7	-0.5	-0.6	-0.8	-1.0	-1.1
8	-0.1	-0.3	-0.4	-0.8	-0.9
9	-0.1	-0.3	-0.3	-0.6	-0.9
10	0.3	-0.1	-0.2	-0.3	-0.5
11	0.4	0.2	0.1	-0.3	-0.3
12	0.5	0.4	0.2	-0.0	-0.2
13	0.7	0.5	0.4	0.2	0.0
14	1.0	0.8	0.6	0.5	0.2
15	1.3	1.0	0.9	0.7	0.5
16	1.3	1.1	0.9	0.7	0.5
17	1.0	1.3	1.0	0.8	0.6
18	1.0	1.2	1.4	1.2	1.0
19	0.8	1.0	1.2	1.4	1.2
20	0.7	0.8	1.1	1.2	1.4

[10]:
```python
# Repeat for carpool
carpool_matrix = pd.DataFrame(index=districts, columns=districts)

for d_from in districts:
    for d_to in districts:
        carpool_matrix.at[d_from, d_to] = utility_carpool(d_from, d_to)

# Convert to plottable format
carpool_matrix = carpool_matrix.astype(float)

plt.imshow(carpool_matrix, cmap='hot', interpolation='nearest')
plt.colorbar()
# Make labels on each column in the middle as ints
plt.xticks(np.arange(0, len(districts), 1), districts)
plt.yticks(np.arange(0, len(districts), 1), districts)
plt.xlabel('Destination')
plt.ylabel('Origin')
plt.title('Carpool utility from district to district')
plt.show()

carpool_matrix.round(1)
```



[10]:

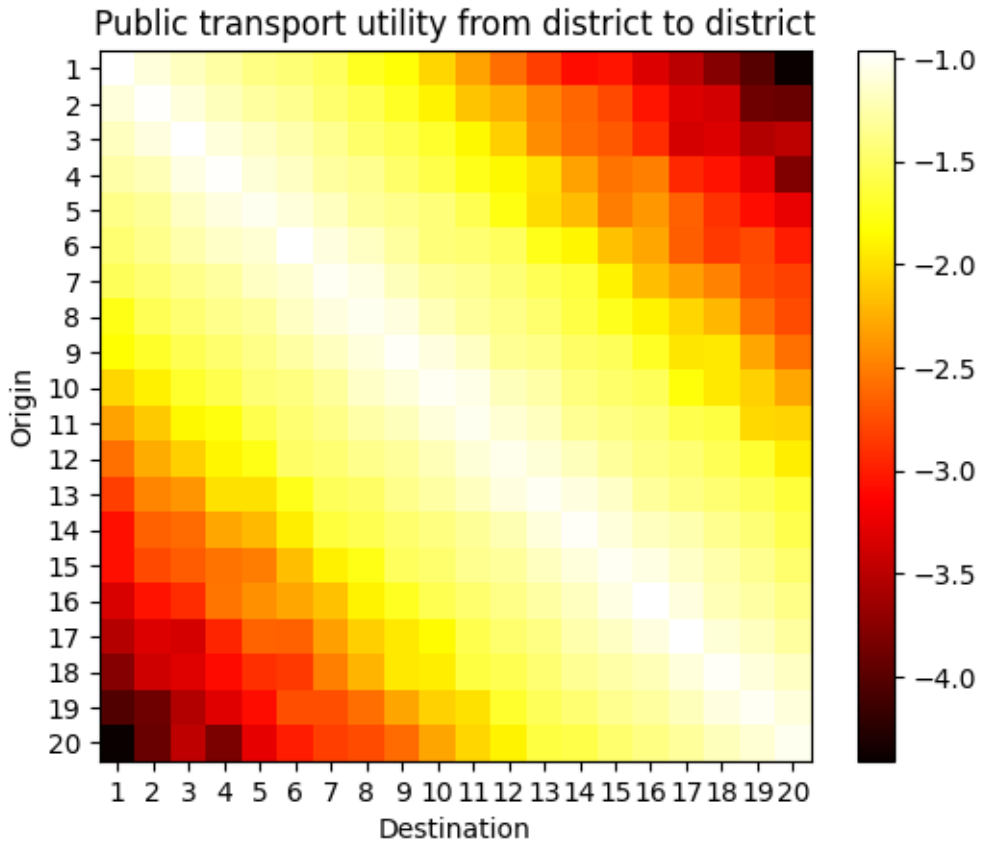
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	\
1	-0.5	-0.7	-0.9	-1.0	-1.2	-1.3	-1.5	-1.7	-1.8	-2.0	-2.2	-2.4	-2.6	-2.8	-2.8	
2	-0.7	-0.6	-0.7	-0.9	-1.1	-1.2	-1.4	-1.5	-1.7	-1.9	-2.0	-2.1	-2.3	-2.5	-2.6	
3	-0.9	-0.7	-0.5	-0.7	-0.9	-1.0	-1.2	-1.4	-1.5	-1.7	-1.8	-2.0	-2.3	-2.4	-2.5	
4	-1.0	-0.9	-0.7	-0.5	-0.7	-0.8	-1.1	-1.2	-1.4	-1.6	-1.7	-1.8	-1.9	-2.2	-2.4	
5	-1.2	-1.1	-0.9	-0.7	-0.6	-0.7	-0.9	-1.1	-1.2	-1.3	-1.5	-1.7	-1.9	-2.1	-2.3	
6	-1.3	-1.2	-1.0	-0.8	-0.7	-0.5	-0.7	-0.9	-1.0	-1.2	-1.3	-1.4	-1.7	-1.9	-2.1	
7	-1.5	-1.4	-1.2	-1.1	-0.9	-0.7	-0.5	-0.7	-0.9	-1.1	-1.2	-1.3	-1.5	-1.6	-1.9	
8	-1.7	-1.5	-1.4	-1.2	-1.1	-0.9	-0.7	-0.6	-0.7	-0.9	-1.0	-1.2	-1.4	-1.6	-1.7	
9	-1.8	-1.7	-1.5	-1.4	-1.2	-1.0	-0.9	-0.7	-0.5	-0.7	-0.9	-1.1	-1.2	-1.4	-1.5	
10	-2.0	-1.9	-1.7	-1.6	-1.3	-1.2	-1.1	-0.9	-0.7	-0.5	-0.7	-0.9	-1.0	-1.2	-1.4	
11	-2.2	-2.0	-1.8	-1.7	-1.5	-1.3	-1.2	-1.0	-0.9	-0.7	-0.6	-0.7	-0.9	-1.1	-1.2	
12	-2.4	-2.1	-2.0	-1.8	-1.7	-1.4	-1.3	-1.2	-1.1	-0.9	-0.7	-0.6	-0.7	-0.9	-1.0	
13	-2.6	-2.3	-2.3	-1.9	-1.9	-1.7	-1.5	-1.4	-1.2	-1.0	-0.9	-0.7	-0.5	-0.7	-0.9	
14	-2.8	-2.5	-2.4	-2.2	-2.1	-1.9	-1.6	-1.6	-1.4	-1.2	-1.1	-0.9	-0.7	-0.6	-0.7	
15	-2.8	-2.6	-2.5	-2.4	-2.3	-2.1	-1.9	-1.7	-1.5	-1.4	-1.2	-1.0	-0.9	-0.7	-0.5	
16	-3.0	-2.8	-2.7	-2.3	-2.2	-2.2	-2.1	-1.9	-1.7	-1.5	-1.4	-1.2	-1.0	-0.9	-0.7	
17	-3.2	-3.0	-3.0	-2.7	-2.5	-2.5	-2.2	-2.0	-1.9	-1.8	-1.6	-1.3	-1.2	-1.0	-0.9	
18	-3.4	-3.1	-3.0	-2.8	-2.7	-2.6	-2.3	-2.1	-1.9	-1.9	-1.6	-1.5	-1.3	-1.2	-1.0	
19	-3.6	-3.5	-3.2	-3.0	-2.8	-2.6	-2.5	-2.4	-2.2	-2.0	-2.0	-1.7	-1.5	-1.3	-1.2	

```
20 -3.9 -3.5 -3.1 -3.4 -3.0 -2.7 -2.6 -2.5 -2.4 -2.2 -2.0 -1.9 -1.6 -1.6 -1.4
```

```
     16   17   18   19   20
1  -3.0 -3.2 -3.4 -3.6 -3.9
2  -2.8 -3.0 -3.1 -3.5 -3.5
3  -2.7 -3.0 -3.0 -3.2 -3.1
4  -2.3 -2.7 -2.8 -3.0 -3.4
5  -2.2 -2.5 -2.7 -2.8 -3.0
6  -2.2 -2.5 -2.6 -2.6 -2.7
7  -2.1 -2.2 -2.3 -2.5 -2.6
8  -1.9 -2.0 -2.1 -2.4 -2.5
9  -1.7 -1.9 -1.9 -2.2 -2.4
10 -1.5 -1.8 -1.9 -2.0 -2.2
11 -1.4 -1.6 -1.6 -2.0 -2.0
12 -1.2 -1.3 -1.5 -1.7 -1.9
13 -1.0 -1.2 -1.3 -1.5 -1.6
14 -0.9 -1.0 -1.2 -1.3 -1.6
15 -0.7 -0.9 -1.0 -1.2 -1.4
16 -0.5 -0.7 -0.9 -1.0 -1.2
17 -0.7 -0.5 -0.7 -0.9 -1.1
18 -0.9 -0.7 -0.6 -0.7 -0.9
19 -1.0 -0.9 -0.7 -0.5 -0.7
20 -1.2 -1.1 -0.9 -0.7 -0.6
```

[11]:
```python
# Repeat for public transport
public_transport_matrix = pd.DataFrame(index=districts, columns=districts)

for d_from in districts:
    for d_to in districts:
        public_transport_matrix.at[d_from, d_to] =␣
 ↪utility_public_transport(d_from, d_to)


# Convert to plottable format
public_transport_matrix = public_transport_matrix.astype(float)

plt.imshow(public_transport_matrix, cmap='hot', interpolation='nearest')
plt.colorbar()
# Make labels on each column in the middle as ints
plt.xticks(np.arange(0, len(districts), 1), districts)
plt.yticks(np.arange(0, len(districts), 1), districts)
plt.xlabel('Destination')
plt.ylabel('Origin')
plt.title('Public transport utility from district to district')
plt.show()

# Show utility matrix to 1 decimal places
```

```
walking_matrix.round(1)
```



[11]:

	1	2	3	4	5	6	7	8	9	10	11	12	\
1	1.2	-1.6	-4.4	-6.9	-9.7	-12.0	-14.3	-18.1	-19.1	-21.9	-25.3	-29.0	
2	-1.6	0.7	-1.7	-4.9	-7.7	-9.7	-12.3	-14.3	-17.8	-20.4	-23.2	-24.8	
3	-4.4	-1.7	1.0	-1.6	-4.4	-6.2	-9.6	-12.3	-15.3	-17.4	-19.8	-22.6	
4	-6.9	-4.9	-1.6	1.2	-1.6	-4.0	-7.5	-9.3	-12.2	-15.6	-18.4	-19.9	
5	-9.7	-7.7	-4.4	-1.6	0.7	-1.6	-4.2	-7.5	-9.3	-11.4	-15.1	-18.5	
6	-12.0	-9.7	-6.2	-4.0	-1.6	1.3	-1.9	-4.2	-7.0	-10.2	-12.0	-13.4	
7	-14.3	-12.3	-9.6	-7.5	-4.2	-1.9	0.9	-1.5	-4.7	-7.6	-9.3	-11.9	
8	-18.1	-14.3	-12.3	-9.3	-7.5	-4.2	-1.5	0.5	-1.8	-4.3	-7.3	-9.5	
9	-19.1	-17.8	-15.3	-12.2	-9.3	-7.0	-4.7	-1.8	1.1	-1.8	-4.4	-7.8	
10	-21.9	-20.4	-17.4	-15.6	-11.4	-10.2	-7.6	-4.3	-1.8	0.7	-1.3	-4.8	
11	-25.3	-23.2	-19.8	-18.4	-15.1	-12.0	-9.3	-7.3	-4.4	-1.3	0.3	-1.8	
12	-29.0	-24.8	-22.6	-19.9	-18.5	-13.4	-11.9	-9.5	-7.8	-4.8	-1.8	0.3	
13	-31.8	-27.6	-26.6	-21.3	-21.4	-18.1	-14.3	-12.5	-9.5	-6.9	-4.4	-2.1	
14	-35.2	-29.7	-29.0	-25.4	-24.0	-20.2	-16.6	-15.5	-12.6	-10.1	-7.7	-4.7	
15	-35.0	-31.5	-30.4	-28.8	-27.8	-23.4	-20.4	-18.4	-14.1	-12.5	-9.3	-7.3	
16	-38.5	-34.9	-33.4	-27.9	-26.5	-25.2	-23.6	-20.3	-17.9	-14.4	-12.1	-9.9	
17	-41.2	-38.2	-39.1	-33.9	-29.8	-29.9	-25.7	-22.5	-21.0	-19.1	-15.3	-12.0	

```
18 -44.3 -39.5 -38.3 -35.3 -32.9 -32.3 -27.7 -24.3 -21.0 -20.6 -16.2 -14.8
19 -47.7 -46.1 -41.3 -38.3 -35.5 -31.3 -31.1 -28.7 -25.2 -22.2 -21.8 -17.2
20 -52.6 -46.3 -40.5 -44.7 -37.7 -34.2 -31.9 -31.3 -29.0 -25.4 -22.1 -20.5
```

```
      13    14    15    16    17    18    19    20
1  -31.8 -35.2 -35.0 -38.5 -41.2 -44.3 -47.7 -52.6
2  -27.6 -29.7 -31.5 -34.9 -38.2 -39.5 -46.1 -46.3
3  -26.6 -29.0 -30.4 -33.4 -39.1 -38.3 -41.3 -40.5
4  -21.3 -25.4 -28.8 -27.9 -33.9 -35.3 -38.3 -44.7
5  -21.4 -24.0 -27.8 -26.5 -29.8 -32.9 -35.5 -37.7
6  -18.1 -20.2 -23.4 -25.2 -29.9 -32.3 -31.3 -34.2
7  -14.3 -16.6 -20.4 -23.6 -25.7 -27.7 -31.1 -31.9
8  -12.5 -15.5 -18.4 -20.3 -22.5 -24.3 -28.7 -31.3
9   -9.5 -12.6 -14.1 -17.9 -21.0 -21.0 -25.2 -29.0
10  -6.9 -10.1 -12.5 -14.4 -19.1 -20.6 -22.2 -25.4
11  -4.4  -7.7  -9.3 -12.1 -15.3 -16.2 -21.8 -22.1
12  -2.1  -4.7  -7.3  -9.9 -12.0 -14.8 -17.2 -20.5
13   0.9  -1.9  -4.2  -7.1 -10.0 -11.9 -14.2 -16.7
14  -1.9   0.6  -1.7  -4.6  -6.8  -9.3 -11.6 -15.8
15  -4.2  -1.7   1.1  -1.3  -4.3  -6.6  -9.0 -12.2
16  -7.1  -4.6  -1.3   1.2  -1.6  -4.9  -6.9 -10.0
17 -10.0  -6.8  -4.3  -1.6   1.5  -2.1  -4.4  -7.4
18 -11.9  -9.3  -6.6  -4.9  -2.1   0.5  -1.6  -4.4
19 -14.2 -11.6  -9.0  -6.9  -4.4  -1.6   0.8  -1.9
20 -16.7 -15.8 -12.2 -10.0  -7.4  -4.4  -1.9   0.4
```

[12]:
```
# As utility is only dependent on destination
# We will just make a ranked list of the destinations

destination_utilities = pd.DataFrame(index=districts, columns=['Utility'])

for d_to in districts:
    # Just calculate from district 1, as it is arbitrary
    destination_utilities.at[d_to, 'Utility'] = destination_utility(1, d_to)

destination_utilities = destination_utilities.astype(float)

destination_utilities = destination_utilities.sort_values(by='Utility',␣
 ↪ascending=False)

destination_utilities
```

[12]:
```
      Utility
12  10.015329
20   9.915485
15   9.872008
14   9.869998
```

```
19    9.802107
5     9.595460
10    9.478228
3     9.388515
1     9.333211
11    9.273902
18    9.230547
6     9.207678
8     9.196709
16    9.128013
7     9.013663
4     8.892813
2     8.841965
9     8.827160
17    8.628432
13    8.612968
```

[13]:
```python
# Calculating conditional mode choice probabilities P_i(m|d)
# P_i(m|d) = exp(V_i(m|d)) / sum(exp(V_i(m|d))) for all m

# Calculate directly on the df, first adding coulumns for utilites for each mode

df_trips['U_walk'] = 0
df_trips['U_bike'] = 0
df_trips['U_car'] = 0
df_trips['U_carpool'] = 0
df_trips['U_public_transport'] = 0

for index, row in df_trips.iterrows():
    d_from = row['ResiZone']
    d_to = row['DestZone']
    df_trips.at[index, 'U_walk'] = utility_walk(d_from, d_to)
    df_trips.at[index, 'U_bike'] = utility_bike(d_from, d_to)
    df_trips.at[index, 'U_car'] = utility_car(d_from, d_to)
    df_trips.at[index, 'U_carpool'] = utility_carpool(d_from, d_to)
    df_trips.at[index, 'U_public_transport'] = utility_public_transport(d_from,␣
 ↪d_to)

# Calculate the sum of the exponentials for each district combination
df_trips['sum_exp'] = df_trips[['U_walk', 'U_bike', 'U_car', 'U_carpool',␣
 ↪'U_public_transport']].apply(lambda x: np.exp(x)).sum(axis=1)

# Calculate the conditional mode choice probabilities
# using formula (7.3)
df_trips['P_walk'] = np.exp(df_trips['U_walk']) / df_trips['sum_exp']
df_trips['P_bike'] = np.exp(df_trips['U_bike']) / df_trips['sum_exp']
df_trips['P_car'] = np.exp(df_trips['U_car']) / df_trips['sum_exp']
```

```
df_trips['P_carpool'] = np.exp(df_trips['U_carpool']) / df_trips['sum_exp']
df_trips['P_public_transport'] = np.exp(df_trips['U_public_transport']) /␣
 ↪df_trips['sum_exp']


# Show to, from and mode choice probabilities
df_trips[['ResiZone', 'DestZone', 'P_walk', 'P_bike', 'P_car', 'P_carpool',␣
 ↪'P_public_transport']]
```

```
/tmp/ipykernel_457140/2480183440.py:15: FutureWarning: Setting an item of
incompatible dtype is deprecated and will raise an error in a future version of
pandas. Value '1.1529323082663732' has dtype incompatible with int64, please
explicitly cast to a compatible dtype first.
  df_trips.at[index, 'U_walk'] = utility_walk(d_from, d_to)
/tmp/ipykernel_457140/2480183440.py:16: FutureWarning: Setting an item of
incompatible dtype is deprecated and will raise an error in a future version of
pandas. Value '1.8264661541331866' has dtype incompatible with int64, please
explicitly cast to a compatible dtype first.
  df_trips.at[index, 'U_bike'] = utility_bike(d_from, d_to)
/tmp/ipykernel_457140/2480183440.py:17: FutureWarning: Setting an item of
incompatible dtype is deprecated and will raise an error in a future version of
pandas. Value '1.3748375923493121' has dtype incompatible with int64, please
explicitly cast to a compatible dtype first.
  df_trips.at[index, 'U_car'] = utility_car(d_from, d_to)
/tmp/ipykernel_457140/2480183440.py:18: FutureWarning: Setting an item of
incompatible dtype is deprecated and will raise an error in a future version of
pandas. Value '-0.5216917307333517' has dtype incompatible with int64, please
explicitly cast to a compatible dtype first.
  df_trips.at[index, 'U_carpool'] = utility_carpool(d_from, d_to)
/tmp/ipykernel_457140/2480183440.py:19: FutureWarning: Setting an item of
incompatible dtype is deprecated and will raise an error in a future version of
pandas. Value '-0.9700761805307809' has dtype incompatible with int64, please
explicitly cast to a compatible dtype first.
  df_trips.at[index, 'U_public_transport'] = utility_public_transport(d_from,
d_to)
```

[13]:

	ResiZone	DestZone	P_walk	P_bike	P_car	P_carpool	\
0	1	1	0.221403	0.434205	0.276411	0.041486	
1	1	2	0.033624	0.265739	0.556988	0.085961	
2	1	3	0.003212	0.101987	0.701987	0.111394	
3	1	4	0.000327	0.036780	0.746382	0.121473	
4	1	5	0.000026	0.011597	0.756872	0.126592	
..	…	…	…	…	…	…	
395	20	16	0.000019	0.009581	0.772257	0.118383	
396	20	17	0.000201	0.028387	0.767196	0.114639	
397	20	18	0.002968	0.095098	0.721727	0.104688	
398	20	19	0.026752	0.236385	0.596384	0.084307	

```
399        20         20  0.137518   0.395029   0.381328    0.052708

     P_public_transport
0              0.026495
1              0.057689
2              0.081420
3              0.095038
4              0.104912
..                  …
395            0.099760
396            0.089577
397            0.075519
398            0.056172
399            0.033417

[400 rows x 7 columns]
```

[14]:
```
# Mode probabilities for someone living in district 1 and going to district 2
df_trips[(df_trips['ResiZone'] == 1) & (df_trips['DestZone'] == 2)][['P_walk',␣
↪'P_bike', 'P_car', 'P_carpool', 'P_public_transport']]
```

[14]:
```
     P_walk    P_bike     P_car  P_carpool  P_public_transport
1  0.033624  0.265739  0.556988   0.085961            0.057689
```

[15]:
```
# Calculate destination choice probabilities P_i(d)
# using formula (7.4)
# P_i(d) = exp(V_i(d) + I(d)) / sum(exp(V_i(d) + I(d))) for all d
# where
# I(d) = mu * ln(sum( exp(V(m|d)/mu) for all m ))

# Calculate I(d) for all districts
df_trips["I"] = 0

for n in districts:
    for d in districts:
        sum_exp = 0
        for m in ["walk", "bike", "car", "carpool", "public_transport"]:
            if m == "walk":
                sum_exp += np.exp(
                    utility_walk(n, d) / params["mu"]
                )
            elif m == "bike":
                sum_exp += np.exp(
                    utility_bike(n, d) / params["mu"]
                )
            elif m == "car":
                sum_exp += np.exp(
```

```
                    utility_car(n, d) / params["mu"]
                )
            elif m == "carpool":
                sum_exp += np.exp(
                    utility_carpool(n, d) / params["mu"]
                )
            elif m == "public_transport":
                sum_exp += np.exp(
                    utility_public_transport(n, d) / params["mu"]
                )

            else:
                raise ValueError("Unknown mode")
        sum_exp = np.log(sum_exp) * params["mu"]

        df_trips.loc[(df_trips["ResiZone"] == n) & (df_trips["DestZone"] == d),␣
↪"I"] = sum_exp

df_trips
```

```
/tmp/ipykernel_457140/2439959489.py:39: FutureWarning: Setting an item of
incompatible dtype is deprecated and will raise an error in a future version of
pandas. Value '2.2975107734194915' has dtype incompatible with int64, please
explicitly cast to a compatible dtype first.
  df_trips.loc[(df_trips["ResiZone"] == n) & (df_trips["DestZone"] == d), "I"] =
sum_exp
```

[15]:

	ResiZone	DestZone	PopResi	EmpResi	PopDest	\
0	1	1	15446.270280	8990.436751	15446.270280	
1	1	2	15446.270280	8990.436751	8431.287835	
2	1	3	15446.270280	8990.436751	13526.411712	
3	1	4	15446.270280	8990.436751	8663.696994	
4	1	5	15446.270280	8990.436751	14782.811654	
..	…	…	…	…	…	
395	20	16	16808.942787	17720.048513	12576.911044	
396	20	17	16808.942787	17720.048513	5608.609071	
397	20	18	16808.942787	17720.048513	1403.393770	
398	20	19	16808.942787	17720.048513	12938.436403	
399	20	20	16808.942787	17720.048513	16808.942787	

	EmpDest	CarStat	Dist	ae	cc	…	U_car	\
0	8990.436751	0.90	0.289223	6.922697	0.242947	…	1.374838	
1	5653.883233	0.90	2.612138	7.759704	2.194196	…	1.172744	
2	9921.381329	0.90	4.929963	7.397353	4.141169	…	0.971093	
3	5979.914469	0.90	7.037586	7.387384	5.911573	…	0.787730	
4	12480.475745	0.90	9.314707	7.908371	7.824354	…	0.589620	
..	…	…	…	…	…	…	…	

```
395   7323.163165     0.99  9.550852  7.605019  8.022716  …  0.659076
396   4747.014808     0.99  7.420676  7.840353  6.233368  …  0.844401
397   9993.609788     0.99  4.944903  7.851580  4.153719  …  1.059793
398  16131.022424     0.99  2.798100  8.199186  2.350404  …  1.246565
399  17720.048513     0.99  0.925344  7.848028  0.777289  …  1.409495

     U_carpool  U_public_transport    sum_exp    P_walk    P_bike     P_car  \
0    -0.521692           -0.970076  14.306367  0.221403  0.434205  0.276411
1    -0.695910           -1.094740   5.800570  0.033624  0.265739  0.556988
2    -0.869747           -1.183205   3.761934  0.003212  0.101987  0.701987
3    -1.027819           -1.273232   2.945408  0.000327  0.036780  0.746382
4    -1.198603           -1.386453   2.382573  0.000026  0.011597  0.756872
..         …                   …         …       …         …         …
395  -1.216314           -1.387473   2.503059  0.000019  0.009581  0.772257
396  -1.056551           -1.303240   3.032582  0.000201  0.028387  0.767196
397  -0.870868           -1.197472   3.998432  0.002968  0.095098  0.721727
398  -0.709858           -1.115894   5.832438  0.026752  0.236385  0.596384
399  -0.569401           -1.025098  10.735862  0.137518  0.395029  0.381328

     P_carpool  P_public_transport         I
0     0.041486            0.026495  2.297511
1     0.085961            0.057689  1.444348
2     0.111394            0.081420  1.088244
3     0.121473            0.095038  0.880044
4     0.126592            0.104912  0.681162
..         …                   …         …
395   0.118383            0.099760  0.741058
396   0.114639            0.089577  0.924786
397   0.104688            0.075519  1.162995
398   0.084307            0.056172  1.468868
399   0.052708            0.033417  2.014908

[400 rows x 25 columns]
```

[16]:
```
for n in districts:
    for d in districts:
        W = destination_utility(n, d)

        sum_exp = 0
        for d_prime in districts:
            W_prime = destination_utility(n, d_prime)
            I_prime = df_trips[(df_trips["ResiZone"] == n) &␣
 ↪(df_trips["DestZone"] == d_prime)]["I"].values[0]
            sum_exp += np.exp(W_prime + I_prime)

        I_nd = df_trips[(df_trips["ResiZone"] == n) & (df_trips["DestZone"] ==␣
 ↪d)]["I"].values[0]
```

```
        df_trips.loc[(df_trips["ResiZone"] == n) & (df_trips["DestZone"] == d),␣
↪"P_dest"] = np.exp(W + I_nd) / sum_exp

df_trips
```

[16]:

	ResiZone	DestZone	PopResi	EmpResi	PopDest	\
0	1	1	15446.270280	8990.436751	15446.270280	
1	1	2	15446.270280	8990.436751	8431.287835	
2	1	3	15446.270280	8990.436751	13526.411712	
3	1	4	15446.270280	8990.436751	8663.696994	
4	1	5	15446.270280	8990.436751	14782.811654	
..	…	…	…	…	…	
395	20	16	16808.942787	17720.048513	12576.911044	
396	20	17	16808.942787	17720.048513	5608.609071	
397	20	18	16808.942787	17720.048513	1403.393770	
398	20	19	16808.942787	17720.048513	12938.436403	
399	20	20	16808.942787	17720.048513	16808.942787	

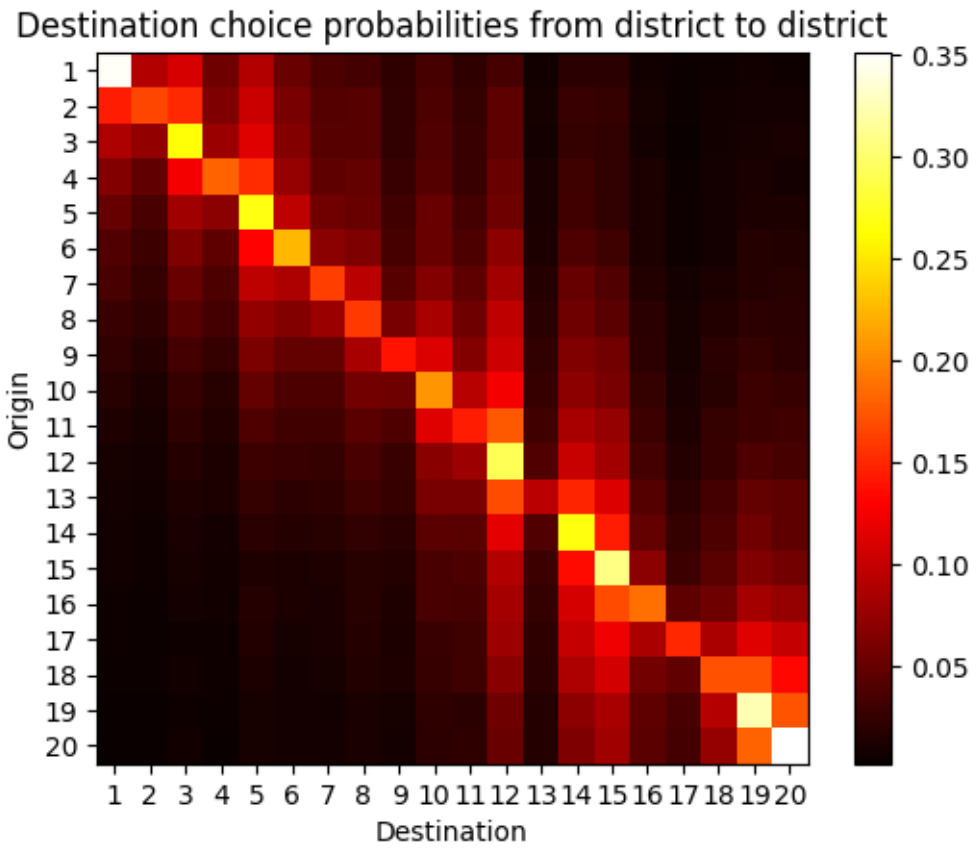
	EmpDest	CarStat	Dist	ae	cc	…	U_carpool	\
0	8990.436751	0.90	0.289223	6.922697	0.242947	…	-0.521692	
1	5653.883233	0.90	2.612138	7.759704	2.194196	…	-0.695910	
2	9921.381329	0.90	4.929963	7.397353	4.141169	…	-0.869747	
3	5979.914469	0.90	7.037586	7.387384	5.911573	…	-1.027819	
4	12480.475745	0.90	9.314707	7.908371	7.824354	…	-1.198603	
..	…	…	…	…	…	…	…	
395	7323.163165	0.99	9.550852	7.605019	8.022716	…	-1.216314	
396	4747.014808	0.99	7.420676	7.840353	6.233368	…	-1.056551	
397	9993.609788	0.99	4.944903	7.851580	4.153719	…	-0.870868	
398	16131.022424	0.99	2.798100	8.199186	2.350404	…	-0.709858	
399	17720.048513	0.99	0.925344	7.848028	0.777289	…	-0.569401	

	U_public_transport	sum_exp	P_walk	P_bike	P_car	P_carpool	\
0	-0.970076	14.306367	0.221403	0.434205	0.276411	0.041486	
1	-1.094740	5.800570	0.033624	0.265739	0.556988	0.085961	
2	-1.183205	3.761934	0.003212	0.101987	0.701987	0.111394	
3	-1.273232	2.945408	0.000327	0.036780	0.746382	0.121473	
4	-1.386453	2.382573	0.000026	0.011597	0.756872	0.126592	
..	…	…	…	…	…	…	
395	-1.387473	2.503059	0.000019	0.009581	0.772257	0.118383	
396	-1.303240	3.032582	0.000201	0.028387	0.767196	0.114639	
397	-1.197472	3.998432	0.002968	0.095098	0.721727	0.104688	
398	-1.115894	5.832438	0.026752	0.236385	0.596384	0.084307	
399	-1.025098	10.735862	0.137518	0.395029	0.381328	0.052708	

	P_public_transport	I	P_dest
0	0.026495	2.297511	0.348079

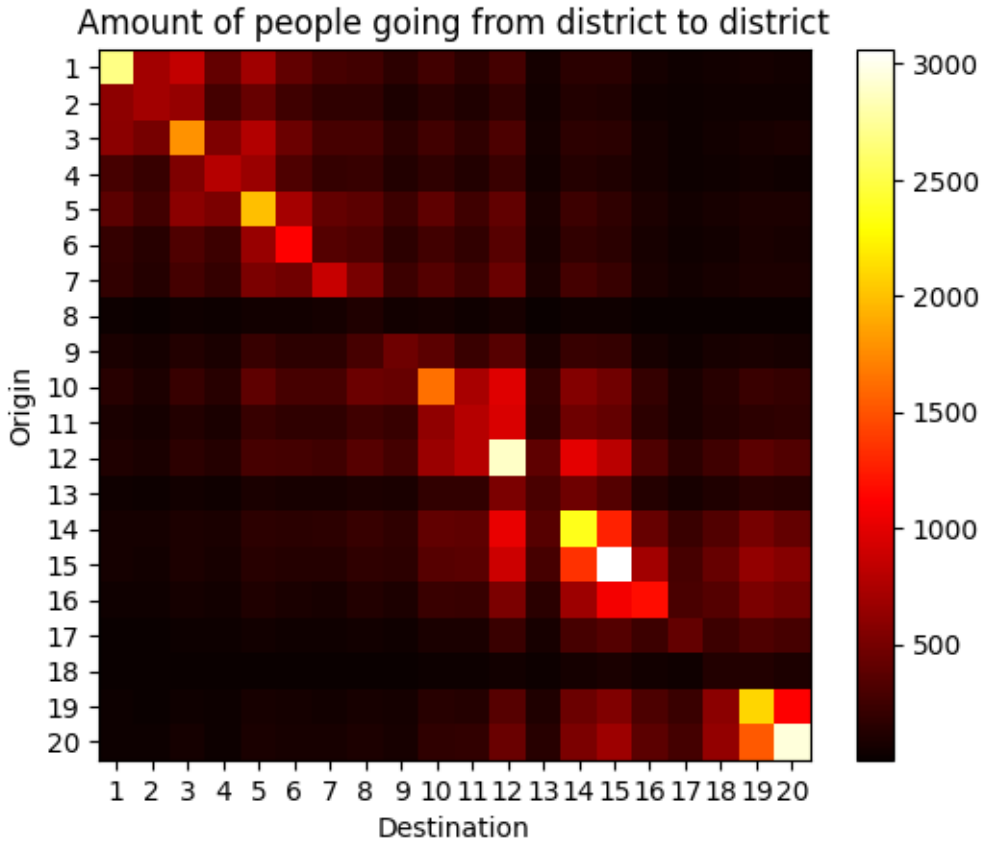
```
1              0.057689  1.444348  0.090742
2              0.081420  1.088244  0.109779
3              0.095038  0.880044  0.054302
4              0.104912  0.681162  0.089867
..                  …         …         …
395            0.099760  0.741058  0.044694
396            0.089577  0.924786  0.032589
397            0.075519  1.162995  0.075513
398            0.056172  1.468868  0.181588
399            0.033417  2.014908  0.351131

[400 rows x 26 columns]
```

[17]:
```
import matplotlib


# Plot P_dest for all districts to see where people from each district are most␣
 ↪likely to go
# Plot as 20x20 matrix heatmap

P_dest_matrix = pd.DataFrame(index=districts, columns=districts)

for d_from in districts:
    for d_to in districts:
        P_dest_matrix.at[d_from, d_to] = df_trips[(df_trips['ResiZone'] ==␣
 ↪d_from) & (df_trips['DestZone'] == d_to)]['P_dest'].values[0]

# Convert to plottable format
P_dest_matrix = P_dest_matrix.astype(float)

plt.imshow(P_dest_matrix, cmap='hot', interpolation='nearest')
plt.colorbar()
# Make labels on each column in the middle as ints
plt.xticks(np.arange(0, len(districts), 1), districts)
plt.yticks(np.arange(0, len(districts), 1), districts)
plt.xlabel('Destination')
plt.ylabel('Origin')
plt.title('Destination choice probabilities from district to district')
plt.show()
```



[18]:
```python
# Calculate the actual amount of people groing from district i to district j
# by combining the destination choice probabilities with the origin zone␣
 ↪population
# row['ResiZone']

travelers = pd.DataFrame(index=districts, columns=districts)

for d_from in districts:
    for d_to in districts:
        # Get the population in the origin zone
        pop = df_trips[df_trips['ResiZone'] == d_from]['PopResi'].values[0]

        # As only half of them are going to work, we divide by 2
        travel_count = pop * 0.5

        travelers.at[d_from, d_to] = travel_count

travelers = travelers.astype(float)

# Multiply the travelers with the destination choice probabilities
```

```
# to get the amount of people going from district i to district j
travelers = travelers * P_dest_matrix


# Convert to plottable format
travelers = travelers.astype(float)

plt.imshow(travelers, cmap='hot', interpolation='nearest')

plt.colorbar()
# Make labels on each column in the middle as ints
plt.xticks(np.arange(0, len(districts), 1), districts)
plt.yticks(np.arange(0, len(districts), 1), districts)
plt.xlabel('Destination')
plt.ylabel('Origin')
plt.title('Amount of people going from district to district')
plt.show()
```



[19]:
```
travelers.sum(axis=1).sum(axis=0)
```

[19]:
```
np.float64(114486.34099588322)
```

[20]:
```
# Make a ranked list of how many people live in each district

populations = pd.DataFrame(index=districts, columns=['Population'])

for d in districts:
    populations.at[d, 'Population'] = df_trips[df_trips['ResiZone'] ==␣
 ↪d]['PopResi'].values[0]

populations = populations.astype(float)

populations = populations.sort_values(by='Population', ascending=False)

populations
```

[20]:

	Population
15	19842.180276
12	19775.635773
14	17670.048518
20	16808.942787
10	15653.849401
1	15446.270280
5	14782.811654
3	13526.411712
19	12938.436403
16	12576.911044
11	10891.448038
7	10744.454211
6	9917.117332
4	8663.696994
2	8431.287835
9	6736.886015
13	6227.373628
17	5608.609071
18	1403.393770
8	1326.917251

[21]:
```
# Calculate by mode how many people are going from district i to district j
# by multiplying the travelers with the mode choice probabilities

walking_amount = pd.DataFrame(index=districts, columns=districts)
biking_amount = pd.DataFrame(index=districts, columns=districts)
car_amount = pd.DataFrame(index=districts, columns=districts)
carpool_amount = pd.DataFrame(index=districts, columns=districts)
public_transport_amount = pd.DataFrame(index=districts, columns=districts)
```

```
for d_from in districts:
    for d_to in districts:
        # Get the amount of people going from district i to district j
        travel_count = travelers.at[d_from, d_to]

        # Get the mode choice probabilities for the route
        P_walk = df_trips[
            (df_trips["ResiZone"] == d_from) & (df_trips["DestZone"] == d_to)
        ]["P_walk"].values[0]
        P_bike = df_trips[
            (df_trips["ResiZone"] == d_from) & (df_trips["DestZone"] == d_to)
        ]["P_bike"].values[0]
        P_car = df_trips[
            (df_trips["ResiZone"] == d_from) & (df_trips["DestZone"] == d_to)
        ]["P_car"].values[0]
        P_carpool = df_trips[
            (df_trips["ResiZone"] == d_from) & (df_trips["DestZone"] == d_to)
        ]["P_carpool"].values[0]
        P_public_transport = df_trips[
            (df_trips["ResiZone"] == d_from) & (df_trips["DestZone"] == d_to)
        ]["P_public_transport"].values[0]

        # Calculate the amount of people going by each mode
        walking_amount.at[d_from, d_to] = travel_count * P_walk
        biking_amount.at[d_from, d_to] = travel_count * P_bike
        car_amount.at[d_from, d_to] = travel_count * P_car
        carpool_amount.at[d_from, d_to] = travel_count * P_carpool
        public_transport_amount.at[d_from, d_to] = travel_count *␣
 ↪P_public_transport

# Sum the amount of people going by each mode
walking_amount = walking_amount.astype(float)
biking_amount = biking_amount.astype(float)
car_amount = car_amount.astype(float)
carpool_amount = carpool_amount.astype(float)
public_transport_amount = public_transport_amount.astype(float)

# Sum the amount on both axis
walking_amount = walking_amount.sum(axis=0).sum(axis=0)
biking_amount = biking_amount.sum(axis=0).sum(axis=0)
car_amount = car_amount.sum(axis=0).sum(axis=0)
carpool_amount = carpool_amount.sum(axis=0).sum(axis=0)
public_transport_amount = public_transport_amount.sum(axis=0).sum(axis=0)

# Pretty print the amount of people going by each mode
print("Amount of people going by each mode:")
print("Walking:", walking_amount.round(0))
```

```
print("Biking:", biking_amount.round(0))
print("Car:", car_amount.round(0))
print("Carpool:", carpool_amount.round(0))
print("Public transport:", public_transport_amount.round(0))
```

```
Amount of people going by each mode:
Walking: 5958.0
Biking: 19565.0
Car: 68606.0
Carpool: 11378.0
Public transport: 8980.0
```

[22]:
```
# Test that the correct amound of people are traveling
sum(populations['Population']) * 0.5, sum([walking_amount, biking_amount,␣
 ↪car_amount, carpool_amount, public_transport_amount])
```

[22]:
```
(114486.34099588325, np.float64(114486.34099588325))
```

[23]:
```
# Calculate market shares for each mode
# Market share = amount of people going by mode / total amount of people␣
 ↪traveling

total_amount = sum([walking_amount, biking_amount, car_amount, carpool_amount,␣
 ↪public_transport_amount])

walking_market_share = walking_amount / total_amount
biking_market_share = biking_amount / total_amount
car_market_share = car_amount / total_amount
carpool_market_share = carpool_amount / total_amount
public_transport_market_share = public_transport_amount / total_amount

# Pretty print the market shares
print("Market shares:")
print("Walking:", walking_market_share.round(3)*100, "%")
print("Biking:", biking_market_share.round(3)*100, "%")
print("Car:", car_market_share.round(3)*100, "%")
print("Carpool:", carpool_market_share.round(3)*100, "%")
print("Public transport:", public_transport_market_share.round(3)*100, "%")
```

```
Market shares:
Walking: 5.2 %
Biking: 17.1 %
Car: 59.9 %
Carpool: 9.9 %
Public transport: 7.8 %
```

[30]:
```
k = 0

# Do all the calculations from above, with given alphas
alphas = [params["k_walk"], params["k_bike"], params["k_car"], params["k_carp"]]


def modelled_market_shares(alphas):
    # Read in the original data
    df = pd.read_excel("PFI_2025_ex2_data.xlsx")

    # Calculate the utility functions for all modes
    df["U_walk"] = 0
    df["U_bike"] = 0
    df["U_car"] = 0
    df["U_carpool"] = 0
    df["U_public_transport"] = 0

    for index, row in df.iterrows():
        d_from = row["ResiZone"]
        d_to = row["DestZone"]
        df.at[index, "U_walk"] = utility_walk(d_from, d_to, alphas[0])
        df.at[index, "U_bike"] = utility_bike(d_from, d_to, alphas[1])
        df.at[index, "U_car"] = utility_car(d_from, d_to, alphas[2])
        df.at[index, "U_carpool"] = utility_carpool(d_from, d_to, alphas[3])
        df.at[index, "U_public_transport"] = utility_public_transport(d_from,␣
 ↪d_to)

    # Calculate the sum of the exponentials for each district combination
    df["sum_exp"] = (
        df[["U_walk", "U_bike", "U_car", "U_carpool", "U_public_transport"]]
        .apply(lambda x: np.exp(x))
        .sum(axis=1)
    )

    # Calculate the conditional mode choice probabilities
    df["P_walk"] = np.exp(df["U_walk"]) / df["sum_exp"]
    df["P_bike"] = np.exp(df["U_bike"]) / df["sum_exp"]
    df["P_car"] = np.exp(df["U_car"]) / df["sum_exp"]
    df["P_carpool"] = np.exp(df["U_carpool"]) / df["sum_exp"]
    df["P_public_transport"] = np.exp(df["U_public_transport"]) / df["sum_exp"]

    # Calculate the destination choice probabilities
    df["I"] = 0

    for n in districts:
        for d in districts:
            sum_exp = 0
```

```
            for m in ["walk", "bike", "car", "carpool", "public_transport"]:
                if m == "walk":
                    sum_exp += np.exp(utility_walk(n, d, alphas[0]) /␣
↪params["mu"])
                elif m == "bike":
                    sum_exp += np.exp(utility_bike(n, d, alphas[1]) /␣
↪params["mu"])
                elif m == "car":
                    sum_exp += np.exp(utility_car(n, d, alphas[2]) /␣
↪params["mu"])
                elif m == "carpool":
                    sum_exp += np.exp(utility_carpool(n, d, alphas[3]) /␣
↪params["mu"])
                elif m == "public_transport":
                    sum_exp += np.exp(utility_public_transport(n, d) /␣
↪params["mu"])
                else:
                    raise ValueError("Unknown mode")
            sum_exp = np.log(sum_exp) * params["mu"]

            df.loc[(df["ResiZone"] == n) & (df["DestZone"] == d), "I"] = sum_exp

    for n in districts:
        for d in districts:
            W = destination_utility(n, d)

            sum_exp = 0
            for d_prime in districts:
                W_prime = destination_utility(n, d_prime)
                I_prime = df[(df["ResiZone"] == n) & (df["DestZone"] ==␣
↪d_prime)][
                    "I"
                ].values[0]
                sum_exp += np.exp(W_prime + I_prime)

            I_nd = df[(df["ResiZone"] == n) & (df["DestZone"] == d)]["I"].
↪values[0]

            df.loc[(df["ResiZone"] == n) & (df["DestZone"] == d), "P_dest"] = (
                np.exp(W + I_nd) / sum_exp
            )

    # Calculate the amount of people groing from district i to district j
    travelers = pd.DataFrame(index=districts, columns=districts)

    for d_from in districts:
```

```
        for d_to in districts:
            # Get the population in the origin zone
            pop = df[df["ResiZone"] == d_from]["PopResi"].values[0]

            # As only half of them are going to work, we divide by 2
            travel_count = pop * 0.5

            travelers.at[d_from, d_to] = travel_count

    travelers = travelers.astype(float)

    # Multiply the travelers with the destination choice probabilities
    # to get the amount of people going from district i to district j
    travelers = travelers * df["P_dest"]

    # Calculate by mode how many people are going from district i to district j
    # by multiplying the travelers with the mode choice probabilities
    walking_amount = pd.DataFrame(index=districts, columns=districts)
    biking_amount = pd.DataFrame(index=districts, columns=districts)
    car_amount = pd.DataFrame(index=districts, columns=districts)
    carpool_amount = pd.DataFrame(index=districts, columns=districts)
    public_transport_amount = pd.DataFrame(index=districts, columns=districts)

    for d_from in districts:
        for d_to in districts:
            # Get the amount of people going from district i to district j
            travel_count = travelers.at[d_from, d_to]

            # Get the mode choice probabilities for the route
            P_walk = df[(df["ResiZone"] == d_from) & (df["DestZone"] == d_to)][
                "P_walk"
            ].values[0]
            P_bike = df[(df["ResiZone"] == d_from) & (df["DestZone"] == d_to)][
                "P_bike"
            ].values[0]
            P_car = df[(df["ResiZone"] == d_from) & (df["DestZone"] == d_to)][
                "P_car"
            ].values[0]
            P_carpool = df[(df["ResiZone"] == d_from) & (df["DestZone"] ==␣
↪d_to)][
                "P_carpool"
            ].values[0]
            P_public_transport = df[
                (df["ResiZone"] == d_from) & (df["DestZone"] == d_to)
            ]["P_public_transport"].values[0]

            # Calculate the amount of people going by each mode
```

```
            walking_amount.at[d_from, d_to] = travel_count * P_walk
            biking_amount.at[d_from, d_to] = travel_count * P_bike
            car_amount.at[d_from, d_to] = travel_count * P_car
            carpool_amount.at[d_from, d_to] = travel_count * P_carpool
            public_transport_amount.at[d_from, d_to] = travel_count *␣
↪P_public_transport

    # Sum the amount of people going by each mode
    walking_amount = walking_amount.astype(float)
    biking_amount = biking_amount.astype(float)
    car_amount = car_amount.astype(float)
    carpool_amount = carpool_amount.astype(float)
    public_transport_amount = public_transport_amount.astype(float)

    # Sum the amount on both axis
    walking_amount = walking_amount.sum(axis=0).sum(axis=0)
    biking_amount = biking_amount.sum(axis=0).sum(axis=0)
    car_amount = car_amount.sum(axis=0).sum(axis=0)
    carpool_amount = carpool_amount.sum(axis=0).sum(axis=0)
    public_transport_amount = public_transport_amount.sum(axis=0).sum(axis=0)

    # Calculate market shares for each mode
    # Market share = amount of people going by mode / total amount of people␣
↪traveling
    total_amount = sum(
        [
            walking_amount,
            biking_amount,
            car_amount,
            carpool_amount,
            public_transport_amount,
        ]
    )

    walking_market_share = walking_amount / total_amount
    biking_market_share = biking_amount / total_amount
    car_market_share = car_amount / total_amount
    carpool_market_share = carpool_amount / total_amount
    public_transport_market_share = public_transport_amount / total_amount

    # Remove the

    return [
        walking_market_share,
        biking_market_share,
        car_market_share,
        carpool_market_share,
```

```
        public_transport_market_share,
    ]


modelled_market_shares(alphas)
```

```
/home/mbg/.local/lib/python3.10/site-packages/openpyxl/styles/stylesheet.py:237:
UserWarning: Workbook contains no default style, apply openpyxl's default
  warn("Workbook contains no default style, apply openpyxl's default")
/tmp/ipykernel_457140/1842779984.py:21: FutureWarning: Setting an item of
incompatible dtype is deprecated and will raise an error in a future version of
pandas. Value '1.1529323082663732' has dtype incompatible with int64, please
explicitly cast to a compatible dtype first.
  df.at[index, "U_walk"] = utility_walk(d_from, d_to, alphas[0])
/tmp/ipykernel_457140/1842779984.py:22: FutureWarning: Setting an item of
incompatible dtype is deprecated and will raise an error in a future version of
pandas. Value '1.8264661541331866' has dtype incompatible with int64, please
explicitly cast to a compatible dtype first.
  df.at[index, "U_bike"] = utility_bike(d_from, d_to, alphas[1])
/tmp/ipykernel_457140/1842779984.py:23: FutureWarning: Setting an item of
incompatible dtype is deprecated and will raise an error in a future version of
pandas. Value '1.3748375923493121' has dtype incompatible with int64, please
explicitly cast to a compatible dtype first.
  df.at[index, "U_car"] = utility_car(d_from, d_to, alphas[2])
/tmp/ipykernel_457140/1842779984.py:24: FutureWarning: Setting an item of
incompatible dtype is deprecated and will raise an error in a future version of
pandas. Value '-0.5216917307333517' has dtype incompatible with int64, please
explicitly cast to a compatible dtype first.
  df.at[index, "U_carpool"] = utility_carpool(d_from, d_to, alphas[3])
/tmp/ipykernel_457140/1842779984.py:25: FutureWarning: Setting an item of
incompatible dtype is deprecated and will raise an error in a future version of
pandas. Value '-0.9700761805307809' has dtype incompatible with int64, please
explicitly cast to a compatible dtype first.
  df.at[index, "U_public_transport"] = utility_public_transport(d_from, d_to)
/tmp/ipykernel_457140/1842779984.py:62: FutureWarning: Setting an item of
incompatible dtype is deprecated and will raise an error in a future version of
pandas. Value '2.2975107734194915' has dtype incompatible with int64, please
explicitly cast to a compatible dtype first.
  df.loc[(df["ResiZone"] == n) & (df["DestZone"] == d), "I"] = sum_exp
```

[30]:
```
[np.float64(0.012488334141399004),
 np.float64(0.054829576477243284),
 np.float64(0.6970082043911014),
 np.float64(0.12971863571996792),
 np.float64(0.10595524927028853)]
```