**Assessing Urban Green Equality Using Vienna’s Open Data Portal**

**Despite their many advantages, accessing nature and green spaces is getting increasingly difficult in highly urbanized areas. Some fear that underserved communities are more exposed to these issues. Here, I propose a data-driven way to explore this.**

[[](https://medium.com/@janosovm?source=post_page-----aa628e0237ad--------------------------------)](https://medium.com/@janosovm?source=post_page-----aa628e0237ad--------------------------------)

[[Towards Data Science](https://towardsdatascience.com/?source=post_page-----aa628e0237ad--------------------------------)](https://towardsdatascience.com/?source=post_page-----aa628e0237ad--------------------------------)

[Milan Janosov](https://medium.com/@janosovm?source=post_page-----aa628e0237ad--------------------------------)

·

Published in

[Towards Data Science](https://towardsdatascience.com/?source=post_page-----aa628e0237ad--------------------------------)

·

11 min read

·

Sep 11, 2023

In particular, I pose an urban development question that has lately been gaining interest across professional circles and local governments — now as green equality. This concept refers to the disparities in people accessing green spaces in different parts of a particular city. Here, I explore its financial dimension and see if there are any clear relationships between the available green area per capita and the economic level of that same urban unit.

I will explore two different spatial resolutions of the city — districts and census districts using Esri Shapefiles provided by the Austrian Government’s Open Data Portal. I will also incorporate tabular statistical data (population and income) into the georeferenced administrative areas. Then, I overlay the administrative areas with an official green area dataset, recording the location of each green space in a geospatial format. Then, I combine this information and quantify each urban district’s total green space per capita size. Finally, I relate each area’s financial status, captured by annual net income, to the green area per capita ratio to see if any patterns emerge.

**1. Data source**

Let’s take a look at the Austrian government’s Open Data Portal [here](https://www.data.gv.at/).

When I was writing this article, the website’s English translation wasn’t really working, so instead of relying on my long-forgotten 12 years of German classes, I used DeepL to navigate across the subpages and thousands of datasets.

Then, I collected a couple of data files, both georeferenced (Esri shapefiles) and simple tabular data, which I will use for the later analysis. The data I collected:

**Boundaries — the administrative boundaries of the following spatial units in Vienna:**

* [The administrative boundaries of Vienna](https://www.data.gv.at/katalog/en/dataset/86c29af5-c52c-423d-8764-82c1bb037936)
* [The administrative boundaries of the 23 districts in Vienna](https://www.data.gv.at/katalog/en/dataset/2ee6b8bf-6292-413c-bb8b-bd22dbb2ad4b)
* [The administrative boundaries of the 250 Census districts in Vienna](https://www.data.gv.at/katalog/en/dataset/e4079286-310c-435a-af2d-64604ba9ade5)

**Land-use — information about the location of green spaces and built-in areas:**

* [Green Belt Vienna City of Vienna](https://www.data.gv.at/katalog/en/dataset/10c7b88b-a708-4e17-a7ef-2c1ce0590377) visualizing the existing and dedicated green belt areas, consisting of 1539 geospatial polygon files enclosing green spaces

**Statistics — data on population and income corresponding to the socio-economical level of an are**a:

* [Population per district](https://www.data.gv.at/katalog/en/dataset/0b1c34b4-06c0-49f3-9379-d86cc8b1c78d), annually recorded since 2002, and stored split based on 5-year age groups, gender, and original nationality
* [Population per census district](https://www.data.gv.at/katalog/en/dataset/15516cf5-09c1-4e69-acf8-3d43183dd026), annually recorded since 2008 and stored split based on three irregular age groups, gender, and origin
* [Average net income](https://www.data.gv.at/katalog/en/dataset/d76c0e8b-c599-4700-8a88-29d0d87e563d#resources) since 2002 in the districts of Vienna, expressed in Euros per employee per annum

Additionally, I stored the downloaded data files in a local folder called data.

**2. Basic data exploration**

**2.1 Administrative boundaries**

First, read and visualize the different shape files containing each administrative boundary level to have a closer grip on the city at hand:

folder = 'data'  
admin\_city = gpd.read\_file(folder + '/LANDESGRENZEOGD')  
admin\_district = gpd.read\_file(folder + '/BEZIRKSGRENZEOGD')  
admin\_census = gpd.read\_file(folder + '/ZAEHLBEZIRKOGD')  
  
display(admin\_city.head(1))  
display(admin\_district.head(1))  
display(admin\_census.head(1))

Here we make a note that the column names BEZNR and ZBEZ, correspond to the District ID and the Census district ID, respectively. Unexpectedly, they are stored/parsed in different formats, numpy.float64 and str:

print(type(admin\_district.BEZNR.iloc[0]))  
print(type(admin\_census.ZBEZ.iloc[0]))pyth

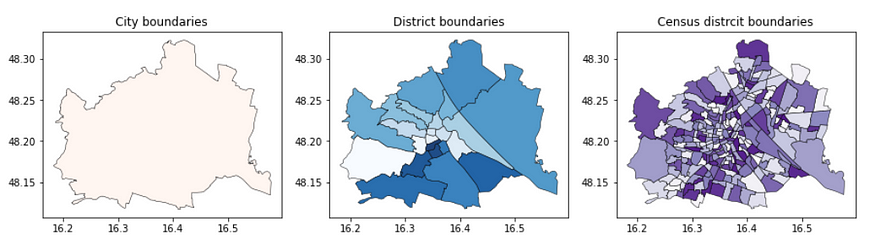
Making sure we indeed have 23 districts and 250 census districts as the data files documentation claimed:

print(len(set(admin\_district.BEZNR)))  
print(len(set(admin\_census.ZBEZ)))

Now visualize the boundaries — first the city, then its districts, and then the even smaller census districts.

f, ax = plt.subplots(1,3,figsize=(15,5))  
  
admin\_city.plot(ax=ax[0],   
 edgecolor = 'k',   
 linewidth = 0.5,   
 alpha = 0.9,   
 cmap = 'Reds')  
  
admin\_district.plot(ax=ax[1],   
 edgecolor = 'k',   
 linewidth = 0.5,   
 alpha = 0.9,   
 cmap = 'Blues')  
  
admin\_census.plot(ax=ax[2],   
 edgecolor = 'k',   
 linewidth = 0.5,   
 alpha = 0.9,   
 cmap = 'Purples')  
  
ax[0].set\_title('City boundaries')  
ax[1].set\_title('District boundaries')  
ax[2].set\_title('Census distrcit boundaries')

This code outputs the following visuals of Vienna:



Differet admin levels of Vienna. Image by the author.

**2.2 Green areas**

Now, also take a look at the green space distribution:

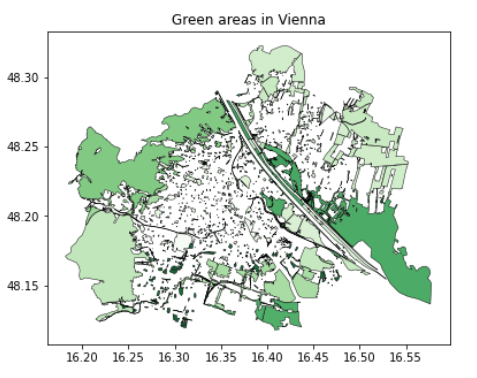
gdf\_green = gpd.read\_file(folder + '/GRUENFREIFLOGD\_GRUENGEWOGD')  
display(gdf\_green.head(3))

Here, one may notice that there is no direct way to link green areas (e.g., no district id-s added) to neighborhoods — so later on, we will do so by manipulating the geometries to find overlaps.

Now visualize this:

f, ax = plt.subplots(1,1,figsize=(7,5))  
  
gdf\_green.plot(ax=ax,   
 edgecolor = 'k',   
 linewidth = 0.5,   
 alpha = 0.9,   
 cmap = 'Greens')  
  
ax.set\_title('Green areas in Vienna')

This code shows where the green areas are within Vienna:



The official green belt of Vienna. Image by the author.

We may note that forestry segments are still within the admin boundary, implying that not every part of the city is urbanized and significantly populated. Later on, we will get back to this when evaluating the per-capital green area.

**2.3 Statistical data — population, income**

Finally, let’s take a look at the statistical data files. The first major difference is that these are not georeferenced but simple csv tables:

df\_pop\_distr = pd.read\_csv('vie-bez-pop-sex-age5-stk-ori-geo4-2002f.csv',   
 sep = ';',  
 encoding='unicode\_escape',   
 skiprows = 1)  
  
df\_pop\_cens = pd.read\_csv('vie-zbz-pop-sex-agr3-stk-ori-geo2-2008f.csv',   
 sep = ';',  
 encoding='unicode\_escape',   
 skiprows = 1)  
  
df\_inc\_distr = pd.read\_csv('vie-bez-biz-ecn-inc-sex-2002f.csv',   
 sep = ';',  
 encoding='unicode\_escape',   
 skiprows = 1)  
  
display(df\_pop\_distr.head(1))  
display(df\_pop\_cens.head(1))  
display(df\_inc\_distr.head(1))

**3. Data preprocessing**

**3.1. Preparing the statistical data files**

The previous subsection shows that the statistical data tables use different naming conventions — they have DISTRICT\_CODE and SUB\_DISTRICT\_CODE identifiers instead of things like BEZNR and ZBEZ. However, after reading each data set’s documentation, it becomes clear that it’s easy to transform from one to another, for which I present two short functions in the next cell. I will simultaneously process data on the level of districts and census districts.

Additionally, I will only be interested in the (latest) aggregated values and data points of the statistical information, such as the total population at the newest snapshot. So, let’s clean up these data files and keep the columns I will use later.

# these functions convert the district and census district ids to be compatbile with the ones found in the shapefiles  
def transform\_district\_id(x):   
 return int(str(x)[1:3])  
  
def transform\_census\_district\_id(x):   
 return int(str(x)[1:5])  
  
# select the latest year of the data set  
df\_pop\_distr\_2 = df\_pop\_distr[df\_pop\_distr.REF\_YEAR \  
 ==max(df\_pop\_distr.REF\_YEAR)]  
df\_pop\_cens\_2 = df\_pop\_cens[df\_pop\_cens.REF\_YEAR \  
 ==max(df\_pop\_cens.REF\_YEAR)]  
df\_inc\_distr\_2 = df\_inc\_distr[df\_inc\_distr.REF\_YEAR \  
 ==max(df\_inc\_distr.REF\_YEAR)]  
  
# convert district ids  
df\_pop\_distr\_2['district\_id'] = \  
 df\_pop\_distr\_2.DISTRICT\_CODE.apply(transform\_district\_id)  
  
df\_pop\_cens\_2['census\_district\_id'] = \  
 df\_pop\_cens\_2.SUB\_DISTRICT\_CODE.apply(transform\_census\_district\_id)  
  
df\_inc\_distr\_2['district\_id'] = \  
 df\_inc\_distr\_2.DISTRICT\_CODE.apply(transform\_district\_id)  
  
  
# aggregate population values  
df\_pop\_distr\_2 = df\_pop\_distr\_2.groupby(by = 'district\_id').sum()  
df\_pop\_distr\_2['district\_population'] = df\_pop\_distr\_2.AUT + \  
 df\_pop\_distr\_2.EEA + df\_pop\_distr\_2.REU + df\_pop\_distr\_2.TCN  
df\_pop\_distr\_2 = df\_pop\_distr\_2[['district\_population']]  
  
df\_pop\_cens\_2 = df\_pop\_cens\_2.groupby(by = 'census\_district\_id').sum()  
df\_pop\_cens\_2['census\_district\_population'] = df\_pop\_cens\_2.AUT \  
 + df\_pop\_cens\_2.FOR  
df\_pop\_cens\_2 = df\_pop\_cens\_2[['census\_district\_population']]  
  
df\_inc\_distr\_2['district\_average\_income'] = \  
 1000\*df\_inc\_distr\_2[['INC\_TOT\_VALUE']]  
df\_inc\_distr\_2 = \  
 df\_inc\_distr\_2.set\_index('district\_id')[['district\_average\_income']]  
  
  
# display the finalized tables  
display(df\_pop\_distr\_2.head(3))  
display(df\_pop\_cens\_2.head(3))  
display(df\_inc\_distr\_2.head(3))  
  
  
# and unifying the naming conventions  
admin\_district['district\_id'] = admin\_district.BEZNR.astype(int)  
admin\_census['census\_district\_id'] = admin\_census.ZBEZ.astype(int)  
  
print(len(set(admin\_census.ZBEZ)))

Double-check the computed total population values at the two levels of aggregations:

print(sum(df\_pop\_distr\_2.district\_population))  
print(sum(df\_pop\_cens\_2.census\_district\_population))

These two should both provide the same result — 1931593 people.

**3.1. Preparing the geospatial data files**

Now that we are done with the essential data preparation of the statistical files, it’s time to match the green area polygons to the administrative area polygons. Then, let’s compute each admin area’s total green area coverage. Additionally, I will add each admin area’s relative green area coverage out of curiosity.

To obtain areas expressed in SI units, we need to switch to a so-called local CRS, which in the case of Vienna is EPSG:31282. You more read more on this topic, map projection and coordinate reference systems [here](https://epsg.io/31282) and [here](https://towardsdatascience.com/how-to-handle-map-projections-properly-in-python-bcbff78895c4).

# converting all GeoDataFrames into the loca crs  
admin\_district\_2 = \  
 admin\_district[['district\_id', 'geometry']].to\_crs(31282)  
  
admin\_census\_2 = \  
 admin\_census[['census\_district\_id', 'geometry']].to\_crs(31282)  
  
gdf\_green\_2 = gdf\_green.to\_crs(31282)

Compute the administrative unit’s area measured in SI units:

admin\_district\_2['admin\_area'] = \  
 admin\_district\_2.geometry.apply(lambda g: g.area)  
  
admin\_census\_2['admin\_area'] = \   
 admin\_census\_2.geometry.apply(lambda g: g.area)  
  
display(admin\_district\_2.head(1))  
display(admin\_census\_2.head(1))

**4. Compute the green area per capita ratio**

4.1 Compute the green area coverage in each administrative unit

I will use GeoPandas’ overlay function to overlay these two administrative boundary GeoDataFrames with the GeoDataFrame containing the green area polygons. Then, I compute the area of each green area section falling into different administrative regions. Next, I sum up these areas to the level of each admin area, both districts and census districts. In the final step, at each resolution unit, I add the administrative previously computed official unit areas and calculate the total area to green area ratio for each district and census district.

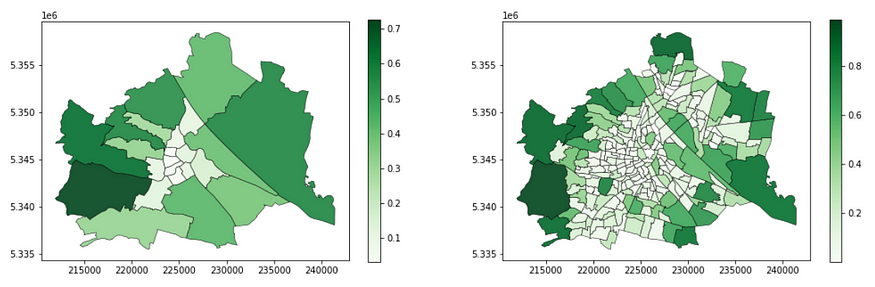
gdf\_green\_mapped\_distr = gpd.overlay(gdf\_green\_2, admin\_district\_2)  
  
gdf\_green\_mapped\_distr['green\_area'] = \  
 gdf\_green\_mapped\_distr.geometry.apply(lambda g: g.area)  
   
gdf\_green\_mapped\_distr = \  
 gdf\_green\_mapped\_distr.groupby(by = 'district\_id').sum()[['green\_area']]  
  
gdf\_green\_mapped\_distr = \  
 gpd.GeoDataFrame(admin\_district\_2.merge(gdf\_green\_mapped\_distr, left\_on = 'district\_id', right\_index = True))  
  
gdf\_green\_mapped\_distr['green\_ratio'] = \  
 gdf\_green\_mapped\_distr.green\_area / gdf\_green\_mapped\_distr.admin\_area  
  
gdf\_green\_mapped\_distr.head(3)

gdf\_green\_mapped\_cens = gpd.overlay(gdf\_green\_2, admin\_census\_2)  
gdf\_green\_mapped\_cens['green\_area'] = \  
 gdf\_green\_mapped\_cens.geometry.apply(lambda g: g.area)  
  
gdf\_green\_mapped\_cens = \  
 gdf\_green\_mapped\_cens.groupby(by = 'census\_district\_id').sum()[['green\_area']]  
  
gdf\_green\_mapped\_cens = \  
 gpd.GeoDataFrame(admin\_census\_2.merge(gdf\_green\_mapped\_cens, left\_on = 'census\_district\_id', right\_index = True))  
  
gdf\_green\_mapped\_cens['green\_ratio'] = gdf\_green\_mapped\_cens.green\_area / gdf\_green\_mapped\_cens.admin\_area  
gdf\_green\_mapped\_cens.head(3)

Finally, visualize the green ratio per district and census district! The results seem to make a lot of sense, with a high level of greenery on the outer parts and much lower in the central areas. Also, the 250 census districts clearly show a more detailed, fine-grained picture of the different neighborhood’s characteristics, offering more profound and more localized insights for urban planners. On the other hand, the district-level information, with ten times fewer spatial units, instead shows grand averages.

f, ax = plt.subplots(1,2,figsize=(17,5))  
  
gdf\_green\_mapped\_distr.plot(ax = ax[0],   
 column = 'green\_ratio',   
 edgecolor = 'k',   
 linewidth = 0.5,   
 alpha = 0.9,   
 legend = True,  
 cmap = 'Greens')  
  
gdf\_green\_mapped\_cens.plot(ax = ax[1],   
 column = 'green\_ratio',   
 edgecolor = 'k',   
 linewidth = 0.5,   
 alpha = 0.9,   
 legend = True,  
 cmap = 'Greens')

This block of code outputs the following maps:



These two maps show the green area ratio in each district / census district. in Vienna. Image by the author.

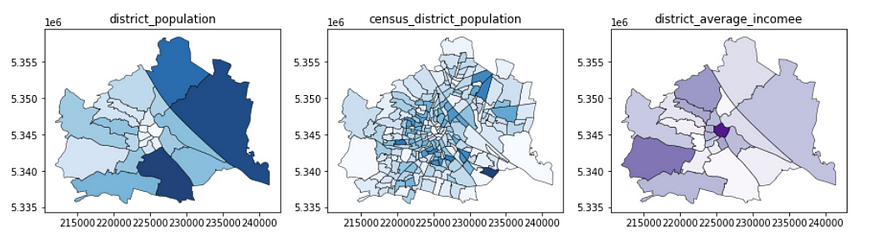
**4.2 Add population and income information for each administrative unit**

In the final step of this section, let’s map the statistical data into administrative areas. Reminder: We have population data on both the level of districts and the level of census districts. However, I could only find income (socioeconomic level indicator) on the level of districts. This is a usual trade-off in geospatial data science. While one dimension (greenery) is much more insightful at the higher resolution (census districts), data constraints may force us to use the lower resolution anyway.

display(admin\_census\_2.head(2))  
display(df\_pop\_cens\_2.head(2))

gdf\_pop\_mapped\_distr = admin\_district\_2.merge(df\_pop\_distr\_2, \  
 left\_on = 'district\_id', right\_index = True)  
  
gdf\_pop\_mapped\_cens = admin\_census\_2.merge(df\_pop\_cens\_2, \  
 left\_on = 'census\_district\_id', right\_index = True)  
  
gdf\_inc\_mapped\_distr = admin\_district\_2.merge(df\_inc\_distr\_2, \  
 left\_on = 'district\_id', right\_index = True)  
  
  
f, ax = plt.subplots(1,3,figsize=(15,5))  
  
gdf\_pop\_mapped\_distr.plot(column = 'district\_population', ax=ax[0], \  
 edgecolor = 'k', linewidth = 0.5, alpha = 0.9, cmap = 'Blues')  
  
gdf\_pop\_mapped\_cens.plot(column = 'census\_district\_population', ax=ax[1], \  
 edgecolor = 'k', linewidth = 0.5, alpha = 0.9, cmap = 'Blues')  
  
gdf\_inc\_mapped\_distr.plot(column = 'district\_average\_income', ax=ax[2], \  
 edgecolor = 'k', linewidth = 0.5, alpha = 0.9, cmap = 'Purples')  
  
ax[0].set\_title('district\_population')  
ax[1].set\_title('census\_district\_population')  
ax[2].set\_title('district\_average\_incomee')

This block of codes results in the following figure:



The various statistical information on Vienna neighborhoods. Image by the author.

**4.3. Green area-per-capita computation**

Let’s sum up what we have now, all integrated into decent shapefiles corresponding to the districts and census districts of Vienna:

On the level of districts, we have green area ratio, population and income data

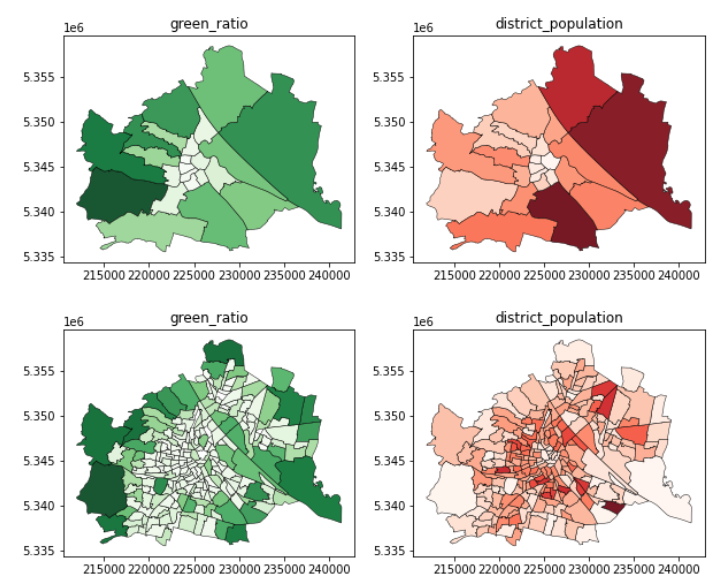
On the level of census districts, we have a green area ratio and population data

To capture green equality simply, I merge the information on the green area’s absolute size and the population in districts and census districts and compute the total amount of green area per capita.

Let’s take a look at our input — green coverage and population:

# a plot for the disticts  
f, ax = plt.subplots(1,2,figsize=(10,5))  
  
gdf\_green\_mapped\_distr.plot(  
 ax = ax[0],   
 column = 'green\_ratio',   
 edgecolor = 'k',   
 linewidth = 0.5,   
 alpha = 0.9,   
 cmap = 'Greens')  
  
gdf\_pop\_mapped\_distr.plot(  
 ax = ax[1],   
 column = 'district\_population',   
 edgecolor = 'k',   
 linewidth = 0.5,   
 alpha = 0.9,   
 cmap = 'Reds')  
  
ax[0].set\_title('green\_ratio')  
ax[1].set\_title('district\_population')  
  
# a plot for the census disticts  
f, ax = plt.subplots(1,2,figsize=(10,5))  
gdf\_green\_mapped\_cens.plot(  
 ax = ax[0],   
 column = 'green\_ratio',   
 edgecolor = 'k',   
 linewidth = 0.5,   
 alpha = 0.9,   
 cmap = 'Greens')  
  
gdf\_pop\_mapped\_cens.plot(  
ax = ax[1],   
 column = 'census\_district\_population',  
 edgecolor = 'k',   
 linewidth = 0.5,   
 alpha = 0.9,   
 cmap = 'Reds')  
  
ax[0].set\_title('green\_ratio')  
ax[1].set\_title('district\_population')

This block of codes results in the following figure:

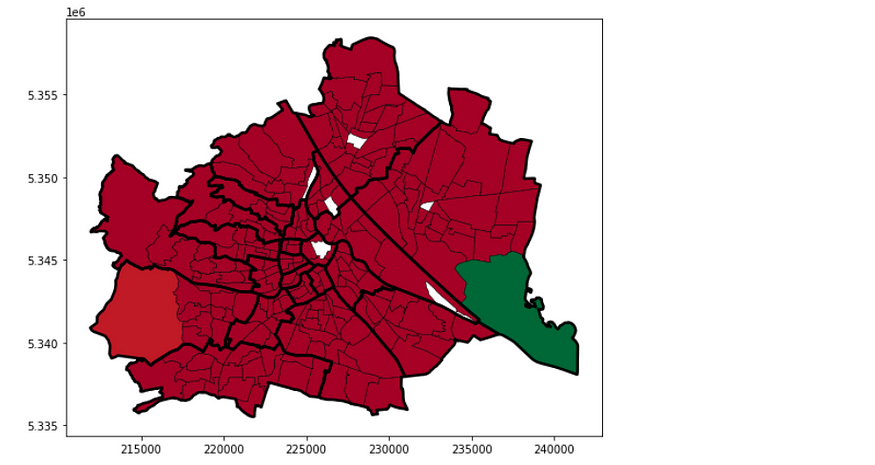


Green area and population levels in Vienna at the levels of districts and census districts. Image by the author.

To compute the green area per capita, I will first merge the greenery and population data frames in the following steps. I will do so via the example of census districts because its higher spatial resolution allows us to observe better patterns (if any) emerging. Make sure we do not divide by zero and also follow common sense; let’s drop those areas that are unpopulated.

gdf\_green\_pop\_cens = \  
 gdf\_green\_mapped\_cens.merge(gdf\_pop\_mapped\_cens.drop( \  
 columns = ['geometry', 'admin\_area']), left\_on = 'census\_district\_id',\  
 right\_on = 'census\_district\_id')[['census\_district\_id', \  
 'green\_area', 'census\_district\_population', 'geometry']]  
  
gdf\_green\_pop\_cens['green\_area\_per\_capita'] = \  
 gdf\_green\_pop\_cens['green\_area'] / \  
 gdf\_green\_pop\_cens['census\_district\_population']  
  
gdf\_green\_pop\_cens = \  
 gdf\_green\_pop\_cens[gdf\_green\_pop\_cens['census\_district\_population']>0]  
  
f, ax = plt.subplots(1,1,figsize=(10,7))  
  
gdf\_green\_pop\_cens.plot(  
 column = 'green\_area\_per\_capita',   
 ax=ax,   
 cmap = 'RdYlGn',   
 edgecolor = 'k',   
 linewidth = 0.5)  
  
admin\_district.to\_crs(31282).plot(\  
 ax=ax, color = 'none', edgecolor = 'k', linewidth = 2.5)

This block of codes results in the following figure:

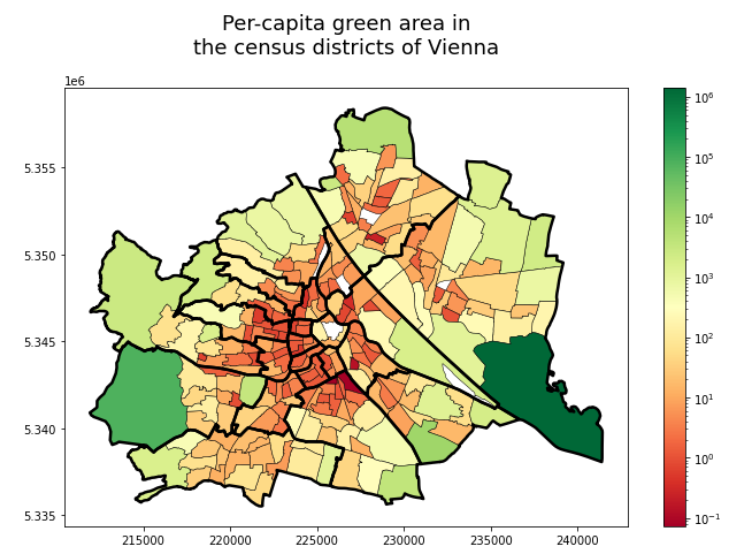


Each census district is colored based on its green area to capita score. Image by the author.

Let’s tweak the visualization a little:

f, ax = plt.subplots(1,1,figsize=(11,7))  
  
ax.set\_title('Per-capita green area in\nthe census districts of Vienna',   
 fontsize = 18, pad = 30)  
  
gdf\_green\_pop\_cens.plot(  
 column = 'green\_area\_per\_capita',   
 ax=ax,   
 cmap = 'RdYlGn',   
 edgecolor = 'k',   
 linewidth = 0.5,   
 legend=True,   
 norm=matplotlib.colors.LogNorm(\  
 vmin=gdf\_green\_pop\_cens.green\_area\_per\_capita.min(), \  
 vmax=gdf\_green\_pop\_cens.green\_area\_per\_capita.max()), )  
  
admin\_district.to\_crs(31282).plot(  
 ax=ax, color = 'none', edgecolor = 'k', linewidth = 2.5)

This block of codes results in the following figure:

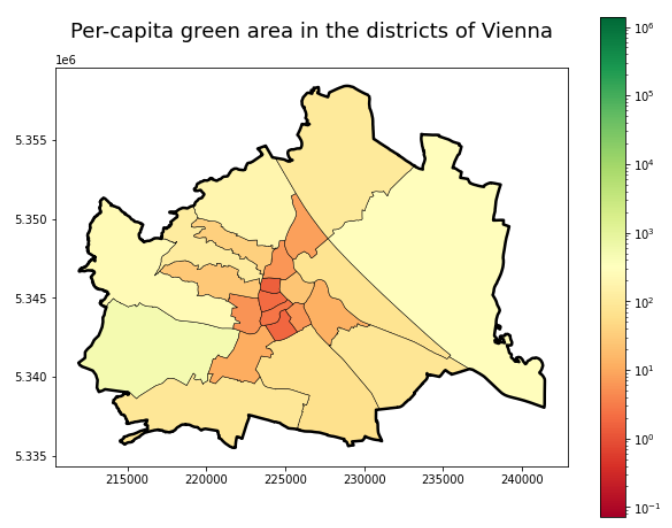


Each census district is colored based on its green area to capita score. Image by the author.

And the same for districts:

# compute the per-capita green area scores  
gdf\_green\_pop\_distr = \  
 gdf\_green\_mapped\_distr.merge(gdf\_pop\_mapped\_distr.drop(columns = \  
 ['geometry', 'admin\_area']), left\_on = 'district\_id', right\_on = \  
 'district\_id')[['district\_id', 'green\_area', 'district\_population', \  
 'geometry']]  
  
gdf\_green\_popdistr = \  
 gdf\_green\_pop\_distr[gdf\_green\_pop\_distr.district\_population>0]  
  
gdf\_green\_pop\_distr['green\_area\_per\_capita'] = \  
 gdf\_green\_pop\_distr['green\_area'] / \  
 gdf\_green\_pop\_distr['district\_population']  
  
# visualize the district-level map  
f, ax = plt.subplots(1,1,figsize=(10,8))  
  
ax.set\_title('Per-capita green area in the districts of Vienna', \   
 fontsize = 18, pad = 26)  
  
gdf\_green\_pop\_distr.plot(column = 'green\_area\_per\_capita', ax=ax, \  
 cmap = 'RdYlGn', edgecolor = 'k', linewidth = 0.5, legend=True, \  
 norm=matplotlib.colors.LogNorm(vmin=\  
 gdf\_green\_pop\_cens.green\_area\_per\_capita.min(), \  
 vmax=gdf\_green\_pop\_cens.green\_area\_per\_capita.max()), )  
  
admin\_city.to\_crs(31282).plot(ax=ax, \  
 color = 'none', edgecolor = 'k', linewidth = 2.5)

This block of codes results in the following figure:



Each district is colored based on its green area to capita score. Image by the author.

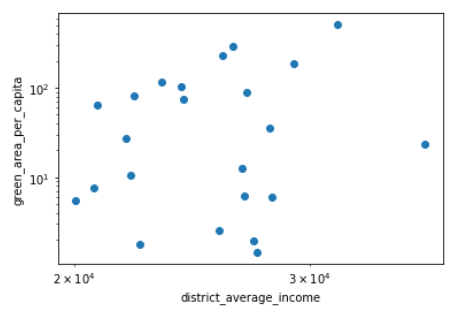
While the significant trends are clear — outer rim, more greenspace for everyone, built-in downtown, reversed. Still, these two plots, especially the more detailed one on the level of census districts, clearly show a variance in the amount of green space people enjoy in the different areas. Further research and incorporating additional data sources, for instance, on land use, could help explain better why those areas are higher in green area or population. For now, let’s enjoy this map and hope everybody finds the right amount of greenery in their home!

# merging the greenery, population and financial data  
gdf\_district\_green\_pip\_inc = \  
 gdf\_green\_pop\_distr.merge(gdf\_inc\_mapped\_distr.drop(columns = \  
 ['geometry']))

Visualize the relationship between the financial and the greenery dimensions:

f, ax = plt.subplots(1,1,figsize=(6,4))  
  
ax.plot(gdf\_district\_green\_pip\_inc.district\_average\_income, \  
 gdf\_district\_green\_pip\_inc.green\_area\_per\_capita, 'o')  
  
ax.set\_xscale('log')  
ax.set\_yscale('log')  
ax.set\_xlabel('district\_average\_income')  
ax.set\_ylabel('green\_area\_per\_capita')

The result of this code block is the following scatter plot:



Constracting the average net income and the green area per capita ratio in the districts of Vienna. Image by the author.

At first glance, the scatterplot doesn’t particularly set a strong case for the financials determining people’s access to green spaces. Honestly, I am a bit surprised by these results — however, in light of Vienna’s conscious, long-standing efforts in greening up their city, it may be why we do not see any major trend here. To confirm, I also checked the correlations between these two variables:

print(spearmanr(gdf\_district\_green\_pip\_inc.district\_average\_income, gdf\_district\_green\_pip\_inc.green\_area\_per\_capita))  
  
print(pearsonr(gdf\_district\_green\_pip\_inc.district\_average\_income, gdf\_district\_green\_pip\_inc.green\_area\_per\_capita))

Due to the heavy-tailed distribution of the financial data, I would take the Spearman (0.13) correlation more seriously here, but even the Pearson correlation (0.30) implies a relatively weak trend, aligning with my previous observations.