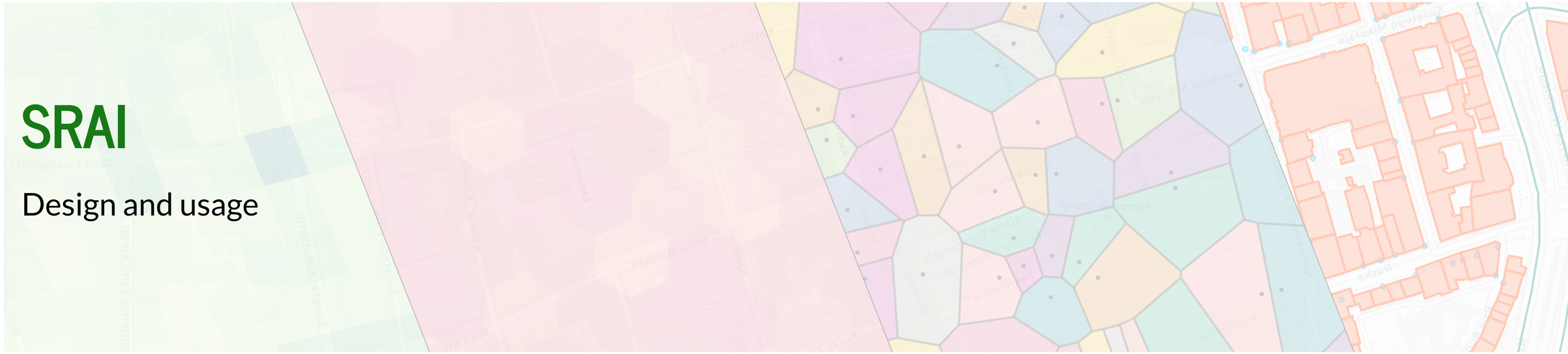
SRAI

Design and usage



What is SRAI?

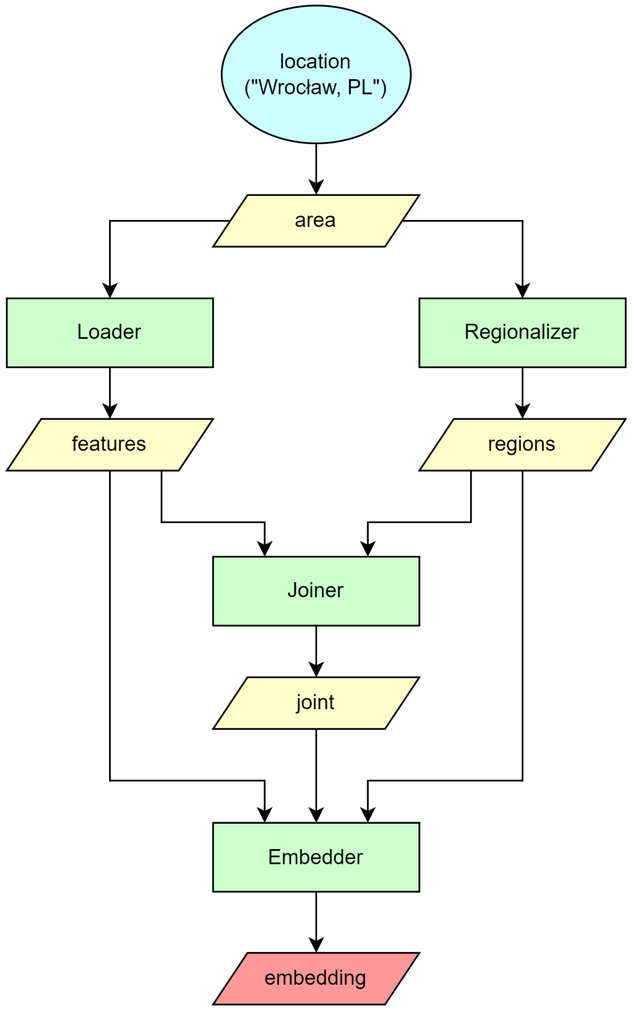
What is SRAI?

A toolbox for geospatial AI

What is SRAI?

A toolbox for geospatial AI

...that aims to standardize the domain


location
("Wrocław, PL")
area
Loader
Regionalizer
features
regions
Joiner
joint
Embedder
embedding

Specify the city

In [3]:
```
CITY = "Basel"
COUNTRY = "Switzerland"

area_name = f"{CITY}, {COUNTRY}"
area_name
```

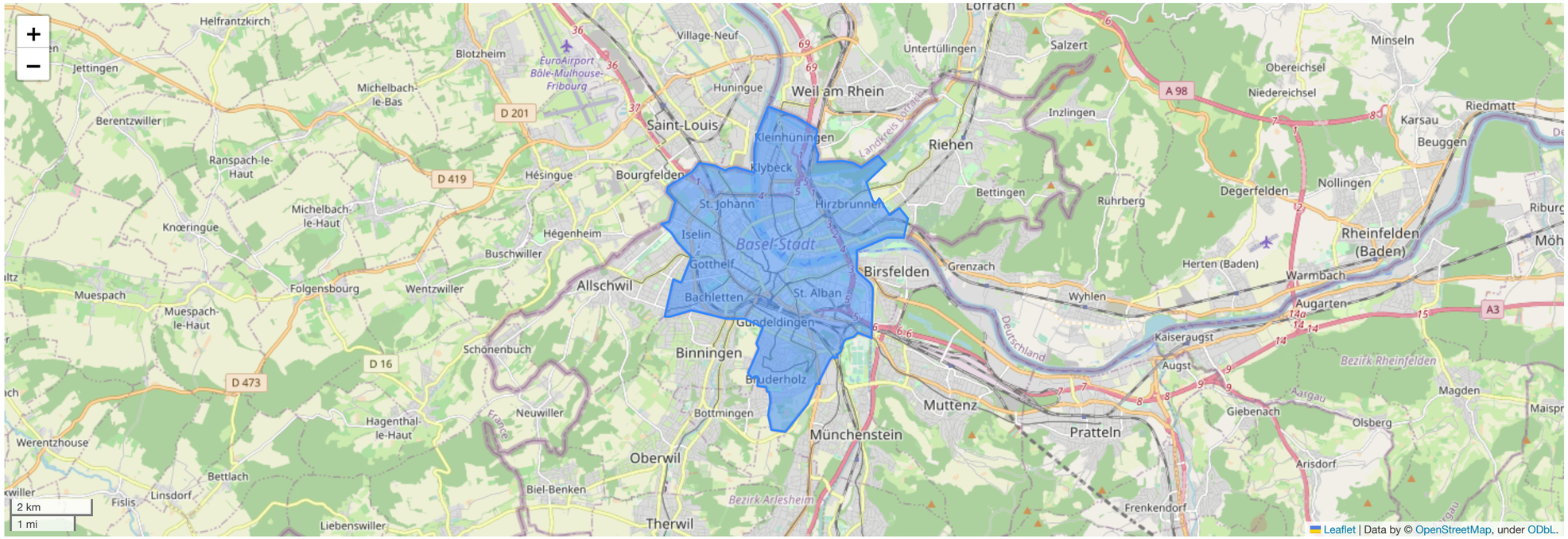
Out[3]:
```
'Basel, Switzerland'
```

Download the area

In [4]:
```
from srai.regionalizers import geocode_to_region_gdf

area = geocode_to_region_gdf(area_name)
area.explore(height=500)
```

Out[4]:



Loaders

Loaders

- used to load spatial data from different sources
- unify loading into a single interface
- prepare data for the embedding methods

API
Examples

Types of loaders:

- GTFS
- OSM Online
- OSM Pbf
- OSM Way
- OSM Tile

Let’s create a city poster

In [5]:
```
from srai.loaders.osm_loaders.filters import BASE_OSM_GROUPS_FILTER

BASE_OSM_GROUPS_FILTER
```

Out[5]:
```
{'water': {'natual': ['water', 'bay', 'beach', 'coastline'],
  'waterway': ['riverbank']},
 'aerialway': {'aerialway': ['cable_car',
   'gondola',
   'mixed_lift',
   'chair_lift',
   'drag_lift',
   't-bar',
   'j-bar',
   'platter',
   'rope_tow',
   'magic_carpet',
   'zip_line',
   'goods',
   'station']},
 'airports': {'aeroway': ['aerodrome', 'heliport', 'spaceport']},
 'sustenance': {'amenity': ['bar',
   'bbq',
   'biergarten',
   'cafe',
   'fast_food',
   'food_court',
   'ice_cream',
   'pub',
   'restaurant']},
 'education': {'amenity': ['college',
   'driving_school',
   'kindergarten',
   'language_school',
   'library',
   'toy_library',
   'music_school',
   'school',
   'university']},
 'transportation': {'amenity': ['bicycle_parking',
   'bicycle_repair_station',
   'bicycle_rental',
   'boat_rental',
   'boat_sharing',
```

In [6]:

```python
from srai.loaders import OSMPbfLoader

features = (
    OSMPbfLoader()
    .load(area, {"highway": True, "water": True, "waterway": True})
    .clip(area)
)
features.head(3)
```

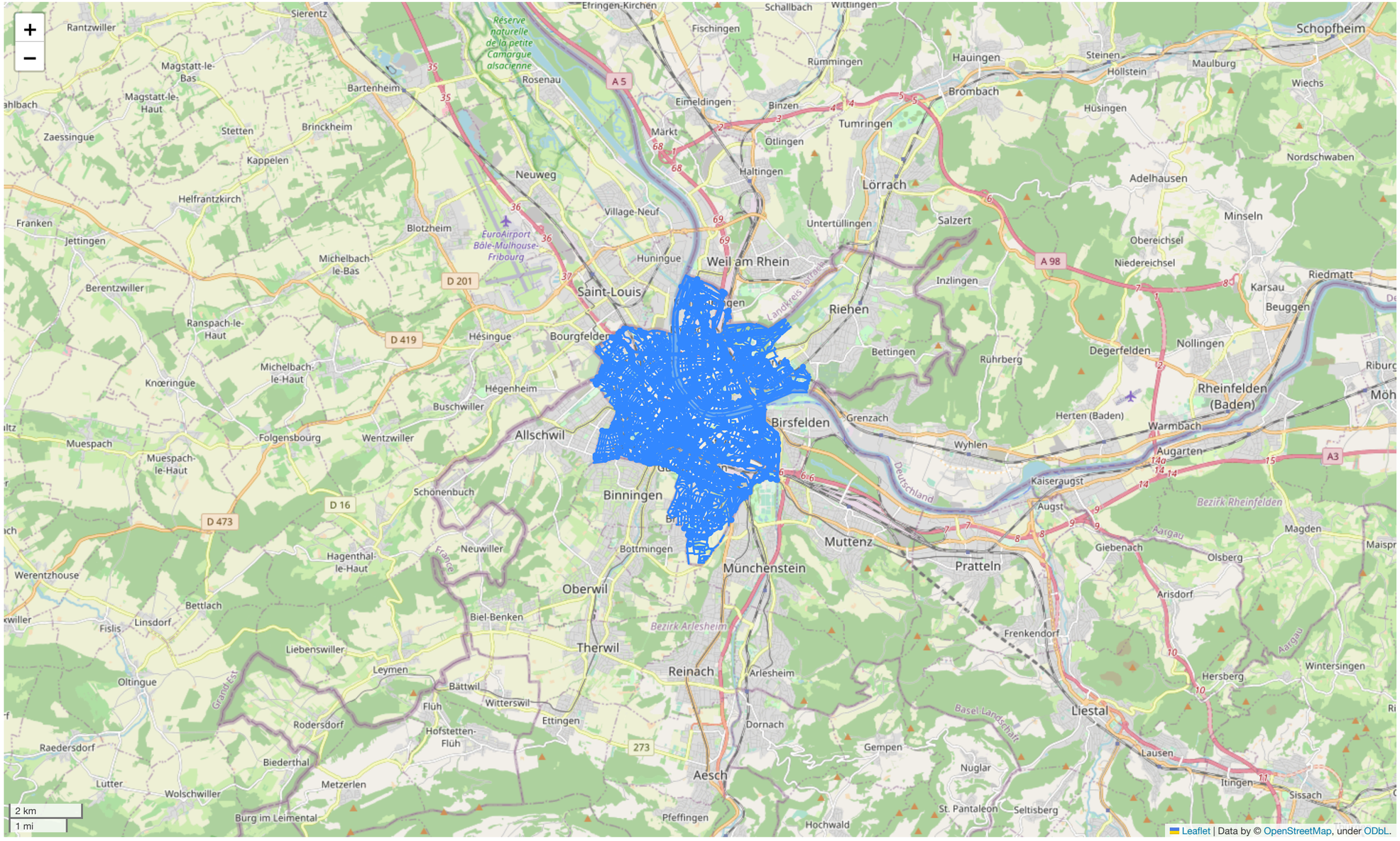
```
[Basel, Basel-City, Switzerland] Counting pbf features: 636246it [00:00, 641996.30it/s]
[Basel, Basel-City, Switzerland] Parsing pbf file #1: 100%|██████████| 636246/636246 [00:02<00:00, 278874.00it/s]
```

Out[6]:

feature_id	geometry	highway	water	waterway
way/26623782	LINESTRING (7.58962 47.52428, 7.58958 47.52412...	track	NaN	NaN
way/23979464	LINESTRING (7.58962 47.52428, 7.58957 47.52428)	footway	NaN	NaN
way/290406242	LINESTRING (7.58964 47.52433, 7.59045 47.52434...	residential	NaN	NaN

In [7]:
```
features.explore()
```

Out[7]:



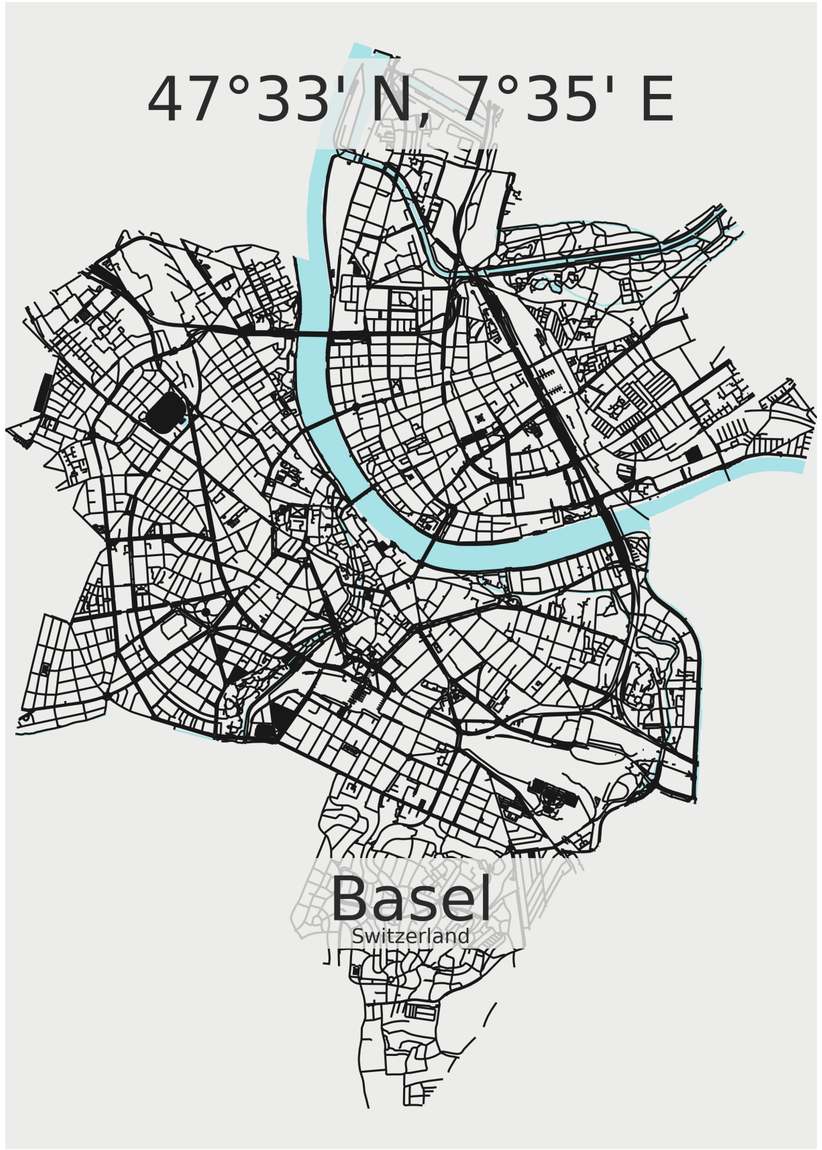
Plot the poster

Plot the poster

In [8]:
```
import matplotlib.pyplot as plt
from utils import plot_poster

ax = plot_poster(features, CITY, COUNTRY)

plt.savefig("poster.png", facecolor="#ecedea", dpi=300)
plt.close()
```


47°33' N, 7°35' E
Basel
Switzerland

GTFS (General Transit Feed Specification)

GTFS (General Transit Feed Specification)

In [9]:
```
from pathlib import Path
from srai.loaders import GTFSLoader
from srai.plotting import plot_regions
```

GTFS (General Transit Feed Specification)

In [9]:
```
from pathlib import Path
from srai.loaders import GTFSLoader
from srai.plotting import plot_regions
```

In [10]:
```
area_vienna = geocode_to_region_gdf("Vienna, Austria")
gtfs_file = Path("./data") / "vienna_gtfs.zip"  # from https://database.mobilitydata.org/

features = GTFSLoader().load(gtfs_file, skip_validation=True).clip(area_vienna)
features.head(3)
```

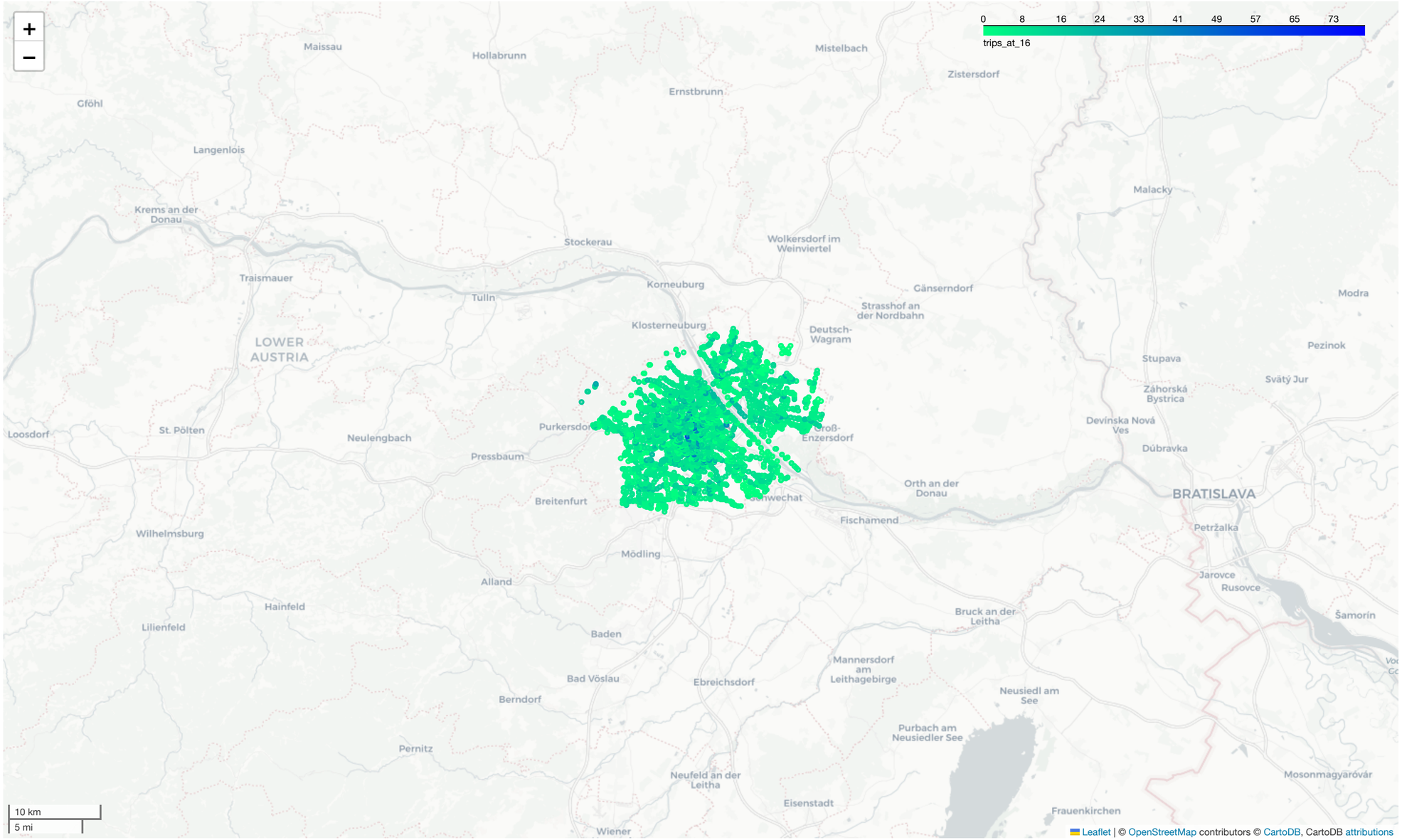
Out[10]:

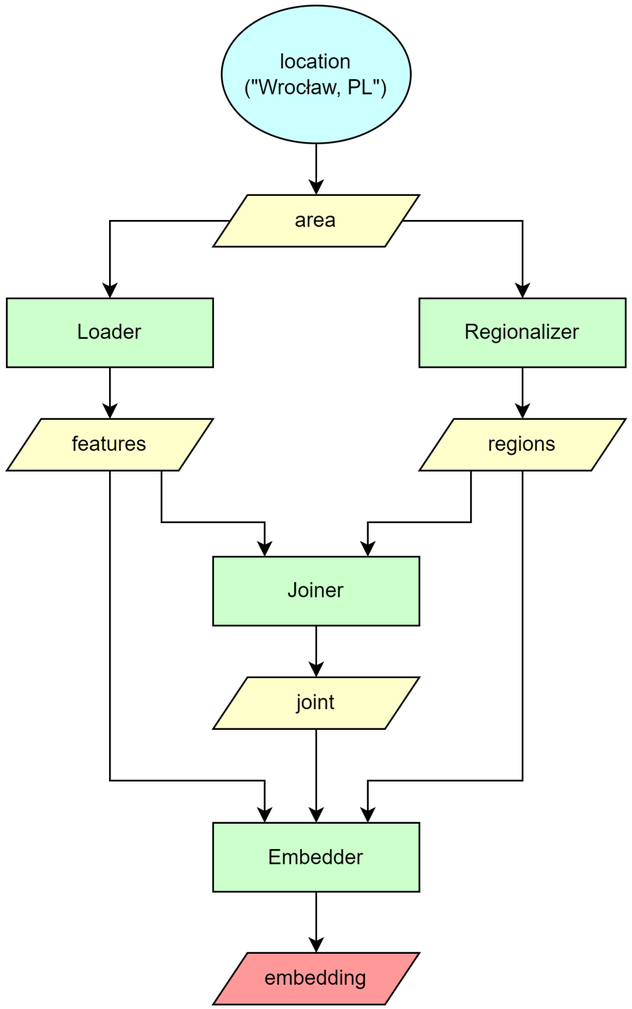
	trips_at_0	trips_at_1	trips_at_2	trips_at_3	trips_at_4	trips_at_5	trips_at_6	trips_at_7	trips_at_8	trips_at_9	...	directions_at_14	directions_at_15	directions_at_16	directions
at:49:789:0:1	2	0	0	0	0	2	4	4	4	4	...	{Wien Liesing}	{Wien Liesing}	{Wien Liesing}	{Wien Lies
at:49:789:0:2	0	0	0	0	0	3	4	4	4	3	...	{Vösendorf-Siebenhirten}	{Vösendorf-Siebenhirten}	{Vösendorf-Siebenhirten}	{Vösendor Siebenhirt
at:49:788:0:1	6	4	2	2	2	10	16	21	22	18	...	{Wien Liesing, Wien Reumannplatz U}	{Wien Liesing, Wien Reumannplatz U}	{Wien Liesing, Wien Reumannplatz U}	{Wien Lies Wien Reumannp U}

3 rows × 49 columns

In [11]:
```
features[["trips_at_16", "geometry"]].explore("trips_at_16", cmap="winter_r", tiles="CartoDB positron")
```

Out[11]:




location
("Wrocław, PL")
area
Loader
Regionalizer
features
regions
Joiner
joint
Embedder
embedding

Regionalizers

Regionalizers

- unify methods for dividing a given area into smaller regions.
- can be based on spatial indexes.

API
Examples

Types of regionalizers:

- H3
- S2
- Voronoi
- Administative Boundary

Spatial index regionalization

Spatial index regionalization

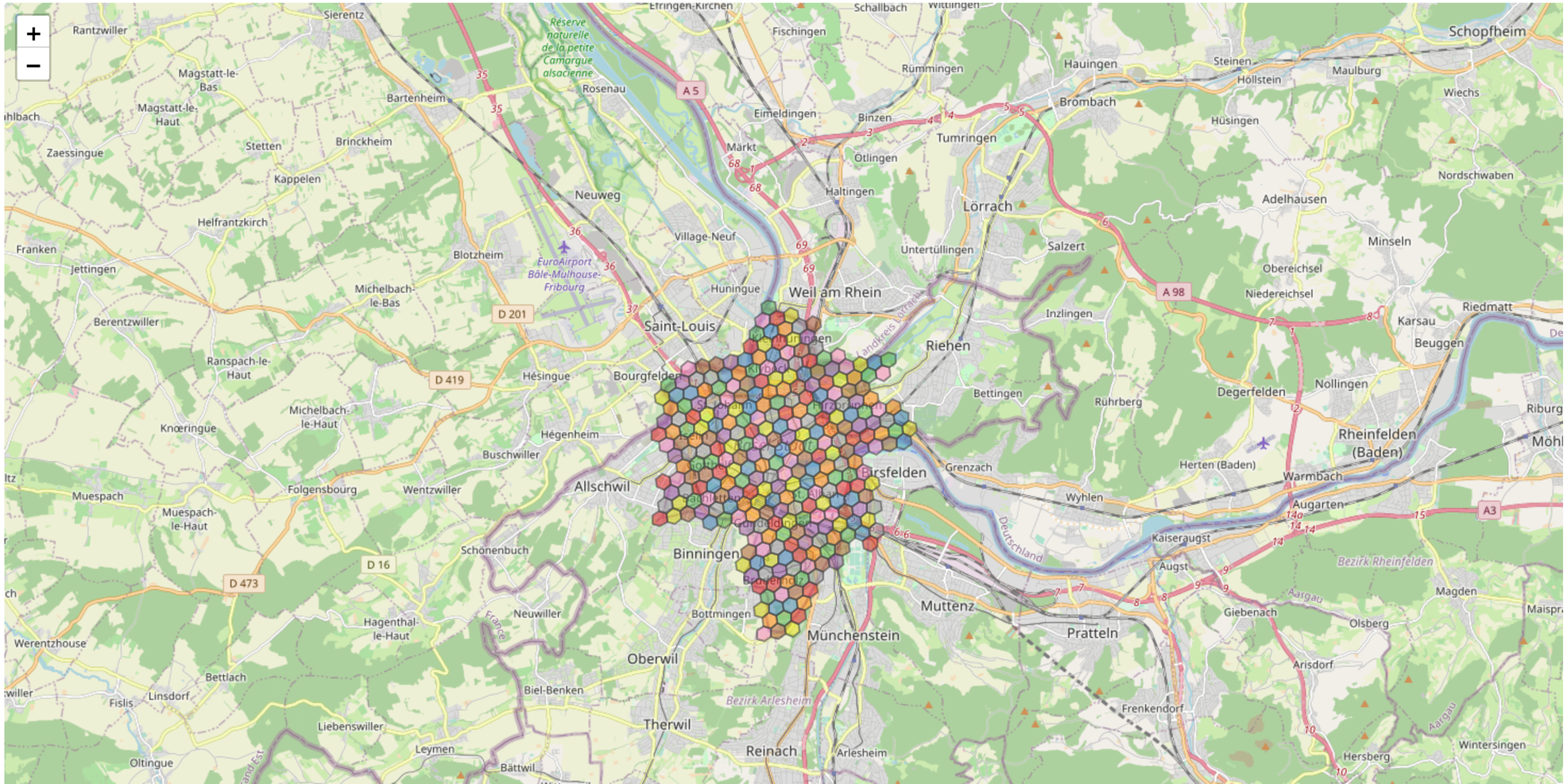
In [12]:
```
from srai.regionalizers import H3Regionalizer
from srai.plotting import plot_regions
```

Spatial index regionalization

In [12]:
```
from srai.regionalizers import H3Regionalizer
from srai.plotting import plot_regions
```

In [13]:
```
regions = H3Regionalizer(resolution=9).transform(area)
plot_regions(regions, colormap=CB_SAFE_PALLETE)
```

Out[13]:



Data driven regionalization

Data driven regionalization

In [14]:
```
from srai.loaders import OSMOnlineLoader
```

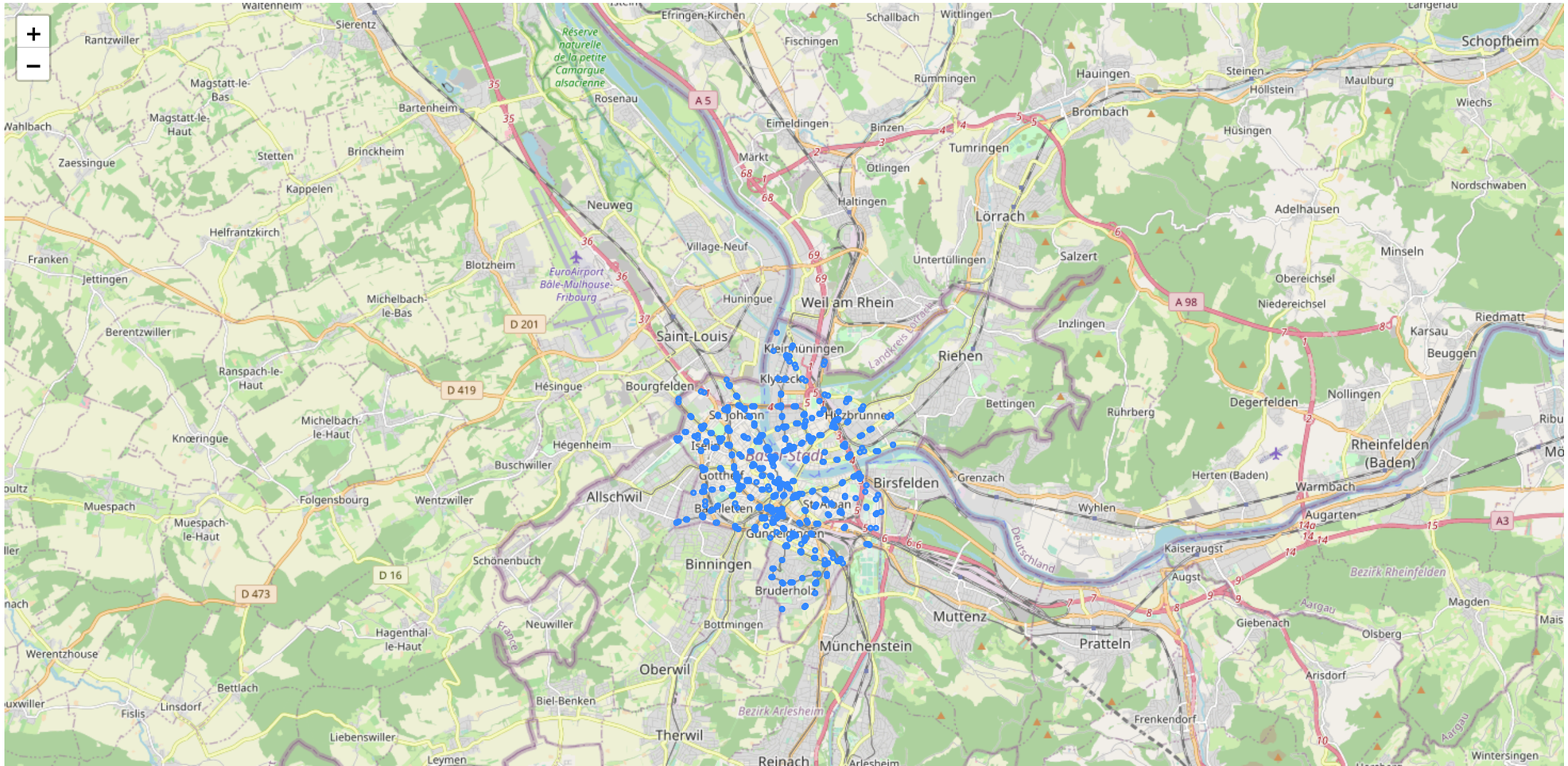
Data driven regionalization

In [14]:
```
from srai.loaders import OSMOnlineLoader
```

In [15]:
```
stops = OSMOnlineLoader().load(area, {"public_transport": "stop_position"})
stops.explore()
```

```
Downloading public_transport: stop_position: 100%|████████████████████████████████████████| 1/1 [00:00<00:00, 13.17it/s]
```

Out[15]:



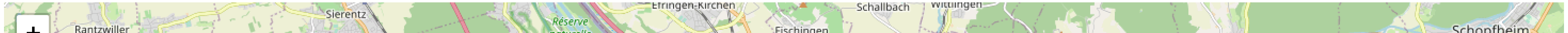
In [16]:
```python
from srai.regionalizers import VoronoiRegionalizer
warnings.simplefilter("ignore")

regionalizer = VoronoiRegionalizer(seeds=stops)
regions = regionalizer.transform(area)

plot_regions(regions, colormap=CB_SAFE_PALLETE)
```

```
Generating regions:   0%|                                                                              | 0/570 [00:00<?, ?it/s]/Us
ers/kacper.lesniara/Projects/Personal/srai-tutorial/venv/lib/python3.10/site-packages/spherical_geometry/great_circle_arc.py:36
5: RuntimeWarning: invalid value encountered in divide
  return P / l
/Users/kacper.lesniara/Projects/Personal/srai-tutorial/venv/lib/python3.10/site-packages/spherical_geometry/great_circle_arc.p
y:365: RuntimeWarning: invalid value encountered in divide
  return P / l
/Users/kacper.lesniara/Projects/Personal/srai-tutorial/venv/lib/python3.10/site-packages/spherical_geometry/great_circle_arc.p
y:365: RuntimeWarning: invalid value encountered in divide
  return P / l
/Users/kacper.lesniara/Projects/Personal/srai-tutorial/venv/lib/python3.10/site-packages/spherical_geometry/great_circle_arc.p
y:365: RuntimeWarning: invalid value encountered in divide
  return P / l
/Users/kacper.lesniara/Projects/Personal/srai-tutorial/venv/lib/python3.10/site-packages/spherical_geometry/great_circle_arc.p
y:365: RuntimeWarning: invalid value encountered in divide
  return P / l
/Users/kacper.lesniara/Projects/Personal/srai-tutorial/venv/lib/python3.10/site-packages/spherical_geometry/great_circle_arc.p
y:365: RuntimeWarning: invalid value encountered in divide
  return P / l
/Users/kacper.lesniara/Projects/Personal/srai-tutorial/venv/lib/python3.10/site-packages/spherical_geometry/great_circle_arc.p
y:365: RuntimeWarning: invalid value encountered in divide
  return P / l
/Users/kacper.lesniara/Projects/Personal/srai-tutorial/venv/lib/python3.10/site-packages/spherical_geometry/great_circle_arc.p
y:365: RuntimeWarning: invalid value encountered in divide
  return P / l
/Users/kacper.lesniara/Projects/Personal/srai-tutorial/venv/lib/python3.10/site-packages/spherical_geometry/great_circle_arc.p
y:365: RuntimeWarning: invalid value encountered in divide
  return P / l
/Users/kacper.lesniara/Projects/Personal/srai-tutorial/venv/lib/python3.10/site-packages/spherical_geometry/great_circle_arc.p
y:365: RuntimeWarning: invalid value encountered in divide
  return P / l
Generating regions: 100%|██████████████████████████████████████████████████████████████████████| 570/570 [00:05<00:00, 97.48it/s]
```

Out[16]:



Administrative boundaries

Administrative boundaries

In [17]:
```
from srai.regionalizers import AdministrativeBoundaryRegionalizer
```

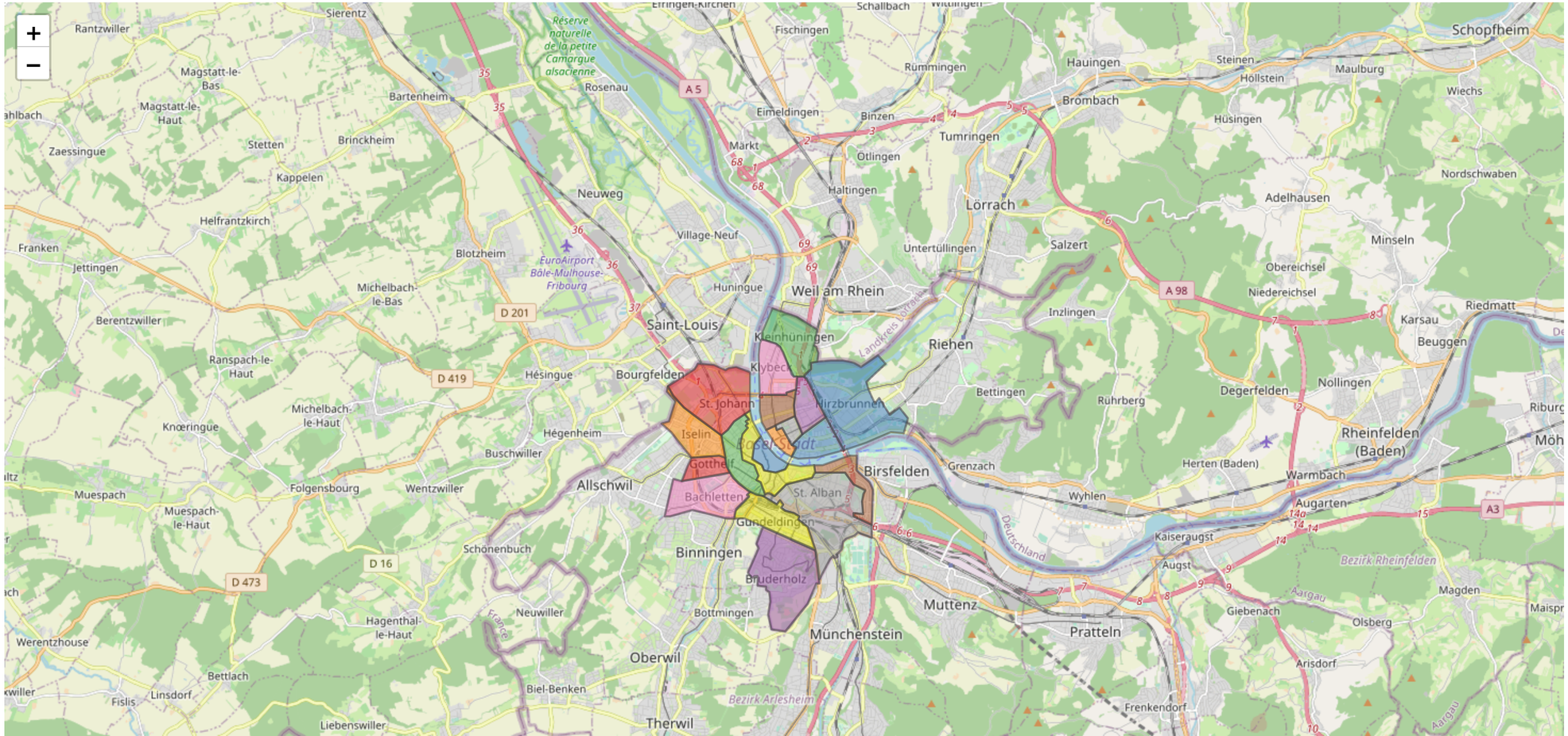
Administrative boundaries

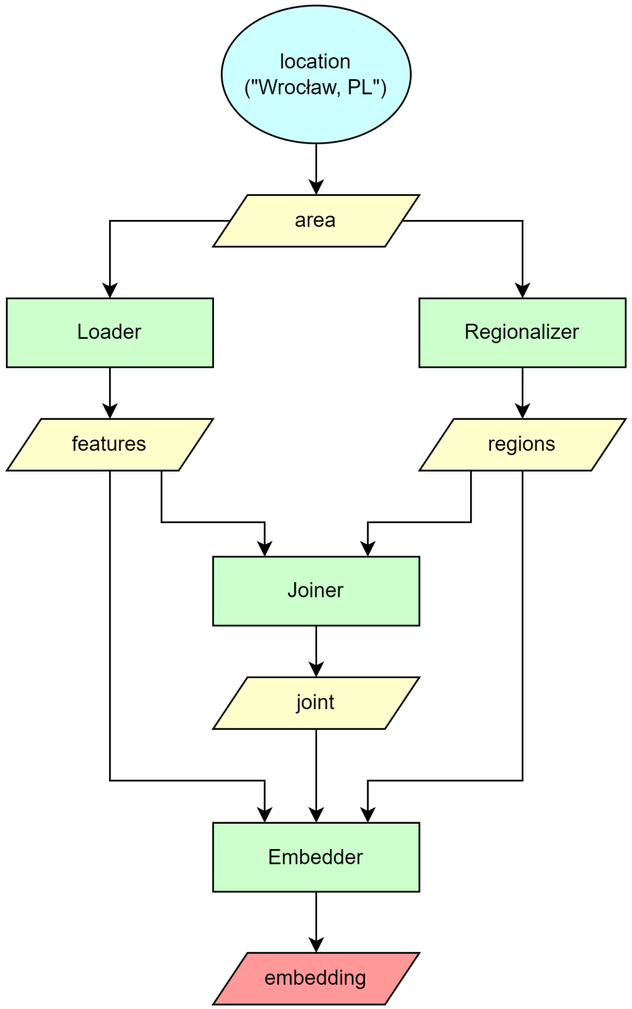
In [17]:
```
from srai.regionalizers import AdministrativeBoundaryRegionalizer
```

In [18]:
```
regionalizer = AdministrativeBoundaryRegionalizer(admin_level=9)
regions = regionalizer.transform(area)
plot_regions(regions, colormap=CB_SAFE_PALLETE)
```

```
Loading boundaries: 20it [00:00, 87.21it/s]
```

Out[18]:




location
("Wrocław, PL")
area
Loader
Regionalizer
features
regions
Joiner
joint
Embedder
embedding

Embedders

Embedders

Unify methods for mapping regions into a vector space.

API
Examples

Types of embedders:

- Count
- Contextual Count
- GTFS2Vec
- Hex2Vec
- Highway2Vec

Simple embeddings

Simple embeddings

In [19]:

```
from srai.loaders import OSMPbfLoader
from srai.regionalizers import H3Regionalizer, geocode_to_region_gdf
from srai.loaders.osm_loaders.filters import HEX2VEC_FILTER

area = geocode_to_region_gdf(area_name)
features = OSMPbfLoader().load(area, HEX2VEC_FILTER)
regions = H3Regionalizer(resolution=9).transform(area)
```

```
[Basel, Basel-City, Switzerland] Counting pbf features: 636246it [00:01, 630452.84it/s]
[Basel, Basel-City, Switzerland] Parsing pbf file #1: 100%|████████████████████| 636246/636246 [00:05<00:00, 107345.88it/s]
```

Simple embeddings

In [19]:
```
from srai.loaders import OSMPbfLoader
from srai.regionalizers import H3Regionalizer, geocode_to_region_gdf
from srai.loaders.osm_loaders.filters import HEX2VEC_FILTER

area = geocode_to_region_gdf(area_name)
features = OSMPbfLoader().load(area, HEX2VEC_FILTER)
regions = H3Regionalizer(resolution=9).transform(area)
```

```
[Basel, Basel-City, Switzerland] Counting pbf features: 636246it [00:01, 630452.84it/s]
[Basel, Basel-City, Switzerland] Parsing pbf file #1: 100%|████████████████████| 636246/636246 [00:05<00:00, 107345.88it/s]
```

In [20]:
```
from srai.joiners import IntersectionJoiner

joint = IntersectionJoiner().transform(regions, features)
```

In [21]:
```
from srai.embedders import CountEmbedder

embedder = CountEmbedder()
embeddings = embedder.transform(regions, features, joint)

embeddings.head(3)
```

Out[21]:

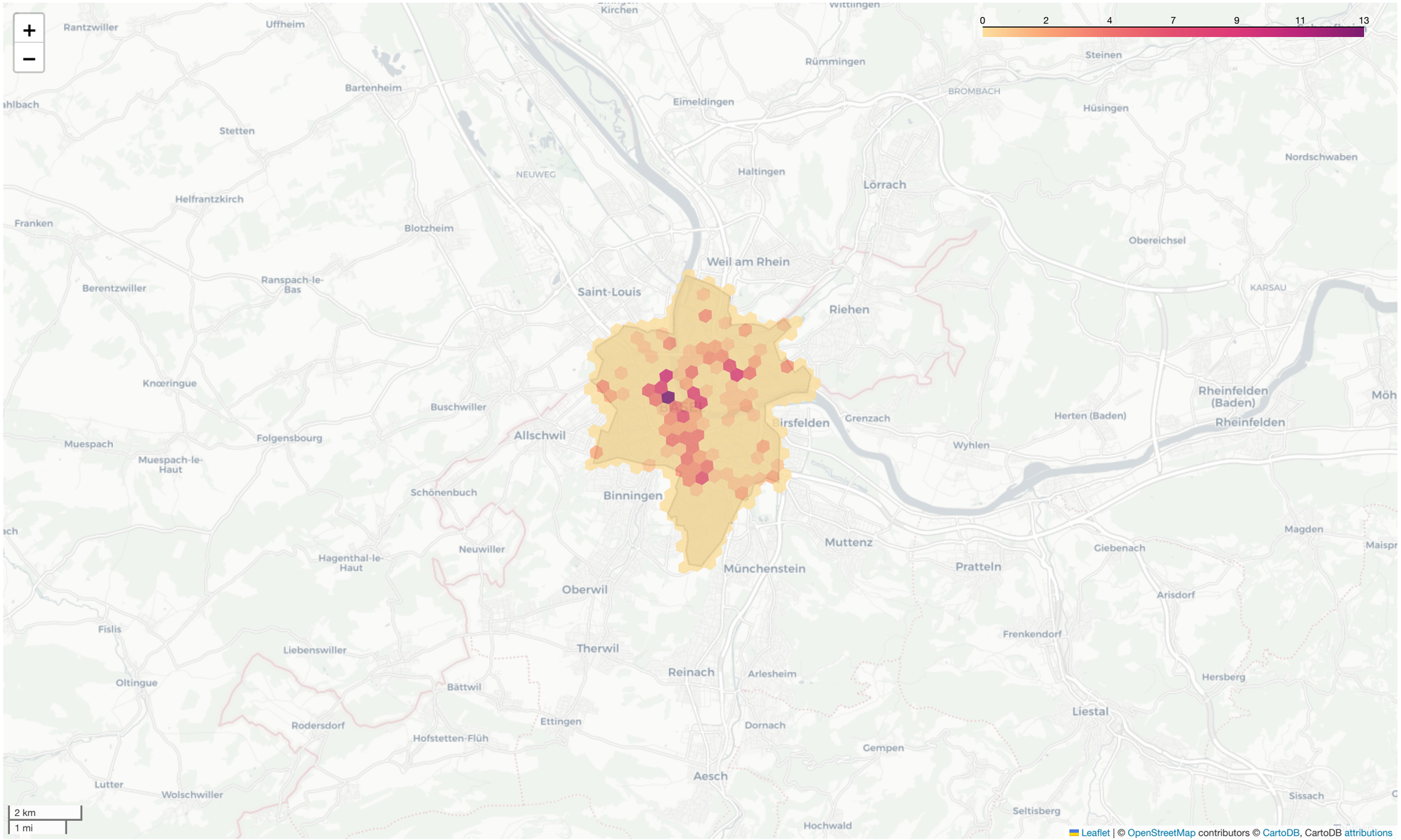
region_id	aeroway_helipad	amenity_arts_centre	amenity_atm	amenity_baby_hatch	amenity_bank	amenity_bar	amenity_bbq	amenity_bench	amenity_bicycle_parking	amenity_bicycle
891f838308fffff	0	0	2	0	1	0	0	4	0	0
891f8383513ffff	0	0	0	0	0	0	0	0	0	0
891f83806dbffff	0	1	0	0	1	0	0	9	1	0

3 rows × 369 columns

In [22]:
```
from srai.plotting import plot_regions, plot_numeric_data

folium_map = plot_regions(area, colormap=["rgba(0,0,0,0.4)"], tiles_style="CartoDB positron")
plot_numeric_data(regions, "amenity_bicycle_parking", embeddings, map=folium_map)
```

Out[22]:



What are we working on?

🏗️ What are we working on?

- Pre-calculated embeddings
- Fine-tuning
- Datasets and benchmarks

Questions?