ADM Project

November 18, 2020

[30]:
```
from collections import defaultdict
import os
from pprint import pprint
from time import time
from IPython.display import HTML
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.style.use('fivethirtyeight')
plt.rcParams['figure.figsize'] = [16, 8]
import seaborn as sns
sns.set_theme(style="darkgrid")
import plotly.express as px
import plotly.graph_objects as go
px.set_mapbox_access_token(os.environ['MAPBOX_TOKEN'])

# db
from sqlalchemy import create_engine
import geoalchemy2
from pymongo import MongoClient, GEOSPHERE
from pymongo.database import Database
from bson.son import SON

# geo
import geojson
import geopandas as gpd
from shapely import wkt
from shapely.geometry import Point, Polygon, MultiPolygon
from shapely.geometry.linestring import LineString
from keplergl import KeplerGl
```

1 Note

1.1 How to create spatial index

```
CREATE INDEX some_descriptive_idx_name
ON table_name
USING GIST (geometry);
```

Note:

The `USING GIST` clause tells PostgreSQL to use the generic index structure (GIST) when building the index. If you receive an error that looks like ERROR: index row requires 11340 bytes, maximum size is 8191 when creating your index, you have likely neglected to add the USING GIST clause.

2 Import data

2.1 Import OpenStreetMap

[2]:
```
# Creating SQLAlchemy's engine to use
username = 'Kai'
password = ''
host = 'localhost'
port = '5432'
database_name = 'test'

engine = create_engine(f'postgresql://{username}:{password}@{host}:{port}/
 ↪{database_name}')
```

[3]:
```
# # read data into geodataframe and load to database
# path = 'data/australia-latest-free.shp'

# for file in os.listdir(path):
#     fn, ext = os.path.splitext(file)

#     if ext == '.shp':
#         table_name = fn.split('_')[2]
#         df = gpd.read_file(os.path.join(path, file))
#         df['geometry'] = df['geometry'].apply(lambda x: WKTElement(x.wkt,␣
 ↪srid=4326))

#         try:
#             df.to_sql(table_name, engine, if_exists='replace', index=False,␣
 ↪dtype={'geometry': Geometry})
#             df.to_postgis(table_name, engine, if_exists='replace')
#             print(f"imported: {table_name:>14}")
#         except:
#             print(f"failed: {table_name:>14}")
```

[4]:
```
%%time
# read data into geodataframe and load to database
path = 'data/australia-latest-free.shp'

for file in os.listdir(path):
    fn, ext = os.path.splitext(file)
```

```
    if ext == '.shp':
        table_name = fn.split('_')[2]

        df = gpd.read_file(os.path.join(path, file))
        df.columns = [col.lower() for col in df.columns]

        try:
            df.to_postgis(table_name, engine, if_exists='replace')
            print(f"imported: {table_name:>14}")
        except:
            print(f"failed: {table_name:>14}")
```

```
imported:        landuse
failed:        natural
imported:      buildings
imported:      waterways
imported:           pofw
imported:          water
imported:      transport
imported:         places
failed:        natural
imported:         places
imported:           pofw
imported:           pois
imported:        traffic
imported:        traffic
imported:       railways
imported:      transport
imported:           pois
imported:          roads
CPU times: user 7min 45s, sys: 35.7 s, total: 8min 21s
Wall time: 11min
```

[5]:
```
!psql -U Kai -d test -c "\d+"
```

```
                        List of relations
 Schema |        Name         | Type  | Owner |   Size    | Description
--------+---------------------+-------+-------+-----------+-------------
 public | airbnb              | table | Kai   | 153 MB    |
 public | airbnb_sydney_100   | table | Kai   | 160 kB    |
 public | airbnb_sydney_100k  | table | Kai   | 145 MB    |
 public | airbnb_sydney_10k   | table | Kai   | 14 MB     |
 public | airbnb_sydney_1k    | table | Kai   | 1480 kB   |
 public | airbnb_sydney_1m    | table | Kai   | 1445 MB   |
 public | airbnb_sydney_500k  | table | Kai   | 723 MB    |
 public | buildings           | table | Kai   | 353 MB    |
 public | districts           | table | Kai   | 92 MB     |
 public | geography_columns   | view  | Kai   | 0 bytes   |
```

```
 public | geometry_columns   | view  | Kai   | 0 bytes  |
 public | landuse            | table | Kai   | 215 MB   |
 public | natural            | table | Kai   | 4088 kB  |
 public | places             | table | Kai   | 2928 kB  |
 public | pofw               | table | Kai   | 1192 kB  |
 public | pois               | table | Kai   | 48 MB    |
 public | railways           | table | Kai   | 10016 kB |
 public | roads              | table | Kai   | 636 MB   |
 public | roads_sydney_100   | table | Kai   | 48 kB    |
 public | roads_sydney_100k  | table | Kai   | 38 MB    |
 public | roads_sydney_10k   | table | Kai   | 3984 kB  |
 public | roads_sydney_1k    | table | Kai   | 432 kB   |
 public | roads_sydney_500k  | table | Kai   | 192 MB   |
 public | spatial_ref_sys    | table | Kai   | 6976 kB  |
 public | sydney_roads       | table | Kai   | 200 MB   |
 public | traffic            | table | Kai   | 23 MB    |
 public | transport          | table | Kai   | 144 kB   |
 public | water              | table | Kai   | 254 MB   |
 public | waterways          | table | Kai   | 178 MB   |
(29 rows)
```

2.2 Import districts

[6]:
```
gpd.read_file('data/australia-latest-free.shp/gis_osm_buildings_a_free_1.shp',␣
↪rows=100).crs
```

[6]:
```
<Geographic 2D CRS: EPSG:4326>
Name: WGS 84
Axis Info [ellipsoidal]:
- Lat[north]: Geodetic latitude (degree)
- Lon[east]: Geodetic longitude (degree)
Area of Use:
- name: World
- bounds: (-180.0, -90.0, 180.0, 90.0)
Datum: World Geodetic System 1984
- Ellipsoid: WGS 84
- Prime Meridian: Greenwich
```

[7]:
```
districts = gpd.read_file('data/nsw_locality_polygon_shp/
↪NSW_LOCALITY_POLYGON_shp.shp')
districts.columns = [col.lower() for col in districts.columns]
districts.crs
```

[7]:
```
<Geographic 2D CRS: EPSG:4283>
Name: GDA94
Axis Info [ellipsoidal]:
```

```
- Lat[north]: Geodetic latitude (degree)
- Lon[east]: Geodetic longitude (degree)
Area of Use:
- name: Australia - GDA
- bounds: (93.41, -60.56, 173.35, -8.47)
Datum: Geocentric Datum of Australia 1994
- Ellipsoid: GRS 1980
- Prime Meridian: Greenwich
```

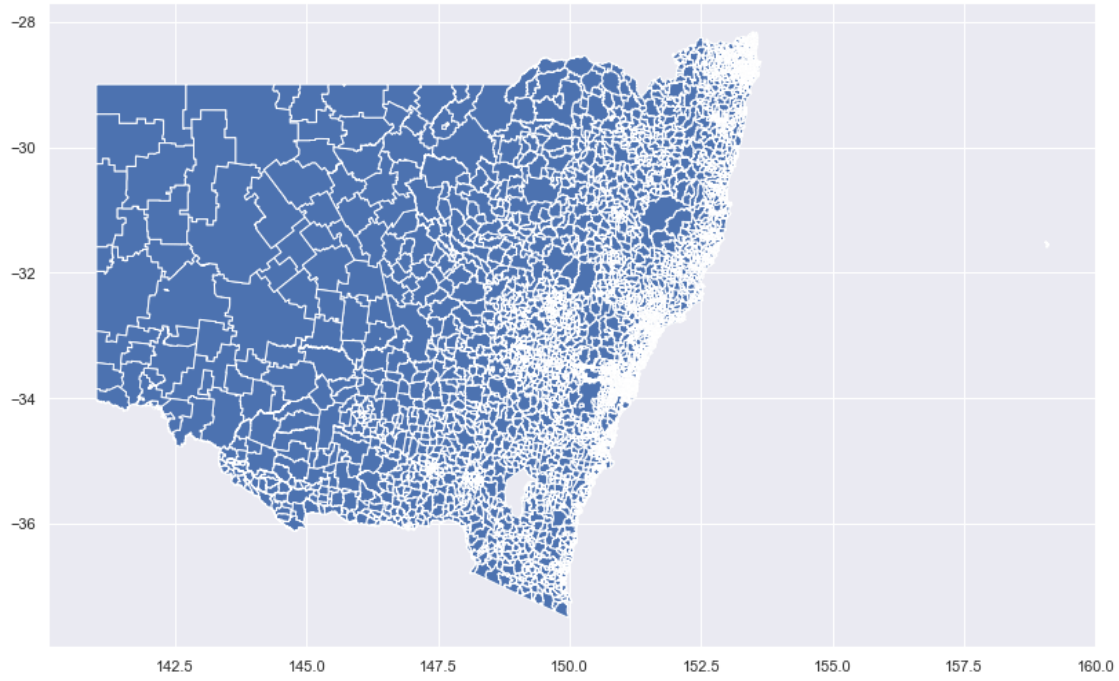
[8]:
```
districts.to_crs(epsg=4326, inplace=True)
districts.crs
```

[8]:
```
<Geographic 2D CRS: EPSG:4326>
Name: WGS 84
Axis Info [ellipsoidal]:
- Lat[north]: Geodetic latitude (degree)
- Lon[east]: Geodetic longitude (degree)
Area of Use:
- name: World
- bounds: (-180.0, -90.0, 180.0, 90.0)
Datum: World Geodetic System 1984
- Ellipsoid: WGS 84
- Prime Meridian: Greenwich
```

[9]:
```
districts.to_postgis('districts', engine, if_exists='replace')
```

[10]:
```
districts.plot()
```

[10]:
```
<AxesSubplot:>
```



[11]:
```
districts.shape
```

[11]: (4591, 13)

[12]:
```
!psql -U Kai -d test -c "\d districts"
```

```
                    Table "public.districts"
   Column   |          Type          | Collation | Nullable | Default
------------+------------------------+-----------+----------+---------
 lc_ply_pid | text                   |           |          |
 dt_create  | text                   |           |          |
 dt_retire  | text                   |           |          |
 loc_pid    | text                   |           |          |
 nsw_locali | text                   |           |          |
 nsw_loca_1 | text                   |           |          |
 nsw_loca_2 | text                   |           |          |
 nsw_loca_3 | text                   |           |          |
 nsw_loca_4 | text                   |           |          |
 nsw_loca_5 | text                   |           |          |
 nsw_loca_6 | text                   |           |          |
 nsw_loca_7 | text                   |           |          |

Indexes:
    "idx_districts_geometry" gist (geometry)
```

2.3 Import airbnb

[13]:
```
df = pd.read_csv('data/airbnb/listings_dec18.csv')
print(df.shape)
df.head()
```

```
/Users/Kai/anaconda3/envs/geo/lib/python3.7/site-
packages/IPython/core/interactiveshell.py:3147: DtypeWarning: Columns
(43,61,62,87) have mixed types.Specify dtype option on import or set
low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

```
(36662, 96)
```

[13]:

	id	listing_url	scrape_id	last_scraped	name	summary	space	description	experiences_offered
0	12351	https://www.airbnb.com/rooms/12351	20181207034750	2018-12-07	Sydney City & Harbour at the door	Come stay with Vinh & Stuart (Awarded as one o…	We're pretty relaxed hosts, and we fully appre…	Come stay with Vinh & Stuart (Awarded as one o…	none
1	14250	https://www.airbnb.com/rooms/14250	20181207034750	2018-12-07	Manly Harbour House	Beautifully renovated, spacious and quiet, our…	Our home is a thirty minute walk along the sea…	Beautifully renovated, spacious and quiet, our…	none
2	15253	https://www.airbnb.com/rooms/15253	20181207034750	2018-12-07	Stunning Penthouse Apartment In Heart Of The City	Penthouse living in a great central location: …	A charming two-level, two-bedroom, two-bathroo…	Penthouse living in a great central location: …	none
3	20865	https://www.airbnb.com/rooms/20865	20181207034750	2018-12-07	3 BED HOUSE + 1 BED STUDIO Balmain	Hi! We are a married professional couple with …	HOUSE : _____ * DUCTED AIR CONDITIONING IN…	Hi! We are a married professional couple with …	none
4	26174	https://www.airbnb.com/rooms/26174	20181207034750	2018-12-07	COZY PRIVATE ROOM, GREAT LOCATION!	NaN	Double bed in decent sized bedroom, in two bed…	Double bed in decent sized bedroom, in two bed…	none

```
                              neighborhood_overview  … requires_license  \
0  Pyrmont is an inner-city village of Sydney, on…  …                 f
1  Balgowlah Heights is one of the most prestigio…  …                 f
2  The location is really central and there is nu…  …                 f
3  BALMAIN is an older inner city village / subur…  …                 f
4                                              NaN  …                 f

  license jurisdiction_names instant_bookable is_business_travel_ready  \
0     NaN               NaN                f                        f
1     NaN               NaN                f                        f
2     NaN               NaN                t                        f
3     NaN               NaN                f                        f
4     NaN               NaN                f                        f

           cancellation_policy  require_guest_profile_picture  \
0  strict_14_with_grace_period                              t
1  strict_14_with_grace_period                              f
2  strict_14_with_grace_period                              f
3  strict_14_with_grace_period                              t
4                     moderate                              f

  require_guest_phone_verification  calculated_host_listings_count  \
0                                t                                2
1                                f                                2
2                                f                                2
3                                t                                1
4                                f                                1

   reviews_per_month
0               4.83
1               0.03
2               3.63
3               0.18
4               0.45

[5 rows x 96 columns]
```

[14]:
```
df.columns
```

[14]:
```
Index(['id', 'listing_url', 'scrape_id', 'last_scraped', 'name', 'summary',
       'space', 'description', 'experiences_offered', 'neighborhood_overview',
       'notes', 'transit', 'access', 'interaction', 'house_rules',
       'thumbnail_url', 'medium_url', 'picture_url', 'xl_picture_url',
       'host_id', 'host_url', 'host_name', 'host_since', 'host_location',
       'host_about', 'host_response_time', 'host_response_rate',
       'host_acceptance_rate', 'host_is_superhost', 'host_thumbnail_url',
```

```
       'host_picture_url', 'host_neighbourhood', 'host_listings_count',
       'host_total_listings_count', 'host_verifications',
       'host_has_profile_pic', 'host_identity_verified', 'street',
       'neighbourhood', 'neighbourhood_cleansed',
       'neighbourhood_group_cleansed', 'city', 'state', 'zipcode', 'market',
       'smart_location', 'country_code', 'country', 'latitude', 'longitude',
       'is_location_exact', 'property_type', 'room_type', 'accommodates',
       'bathrooms', 'bedrooms', 'beds', 'bed_type', 'amenities', 'square_feet',
       'price', 'weekly_price', 'monthly_price', 'security_deposit',
       'cleaning_fee', 'guests_included', 'extra_people', 'minimum_nights',
       'maximum_nights', 'calendar_updated', 'has_availability',
       'availability_30', 'availability_60', 'availability_90',
       'availability_365', 'calendar_last_scraped', 'number_of_reviews',
       'first_review', 'last_review', 'review_scores_rating',
       'review_scores_accuracy', 'review_scores_cleanliness',
       'review_scores_checkin', 'review_scores_communication',
       'review_scores_location', 'review_scores_value', 'requires_license',
       'license', 'jurisdiction_names', 'instant_bookable',
       'is_business_travel_ready', 'cancellation_policy',
       'require_guest_profile_picture', 'require_guest_phone_verification',
       'calculated_host_listings_count', 'reviews_per_month'],
      dtype='object')
```

[15]:
```
cols = ['id', 'listing_url', 'scrape_id', 'last_scraped', 'name', 'summary',
       'space', 'description', 'experiences_offered', 'neighborhood_overview',
       'notes', 'transit', 'access', 'interaction', 'house_rules',
       'thumbnail_url', 'medium_url', 'picture_url', 'xl_picture_url',
       'host_id', 'host_url', 'host_name', 'host_since', 'host_location',
       'host_about', 'host_response_time', 'host_response_rate',
       'host_acceptance_rate', 'host_is_superhost', 'host_thumbnail_url',
       'host_picture_url', 'host_neighbourhood', 'host_listings_count',
       'host_total_listings_count', 'host_verifications',
       'host_has_profile_pic', 'host_identity_verified', 'street',
       'neighbourhood', 'neighbourhood_cleansed',
       'neighbourhood_group_cleansed', 'city', 'state', 'zipcode', 'market',
       'smart_location', 'country_code', 'country', 'latitude', 'longitude',
       'is_location_exact', 'property_type', 'room_type', 'accommodates',
       'bathrooms', 'bedrooms', 'beds', 'bed_type', 'amenities', 'square_feet',
       'price', 'weekly_price', 'monthly_price', 'security_deposit',
       'cleaning_fee', 'guests_included', 'extra_people', 'minimum_nights',
       'maximum_nights', 'calendar_updated', 'has_availability',
       'availability_30', 'availability_60', 'availability_90',
       'availability_365', 'calendar_last_scraped', 'number_of_reviews',
       'first_review', 'last_review', 'review_scores_rating',
       'review_scores_accuracy', 'review_scores_cleanliness',
       'review_scores_checkin', 'review_scores_communication',
       'review_scores_location', 'review_scores_value', 'requires_license',
```

```
        'license', 'jurisdiction_names', 'instant_bookable',
        'is_business_travel_ready', 'cancellation_policy',
        'require_guest_profile_picture', 'require_guest_phone_verification',
        'calculated_host_listings_count', 'reviews_per_month']
```

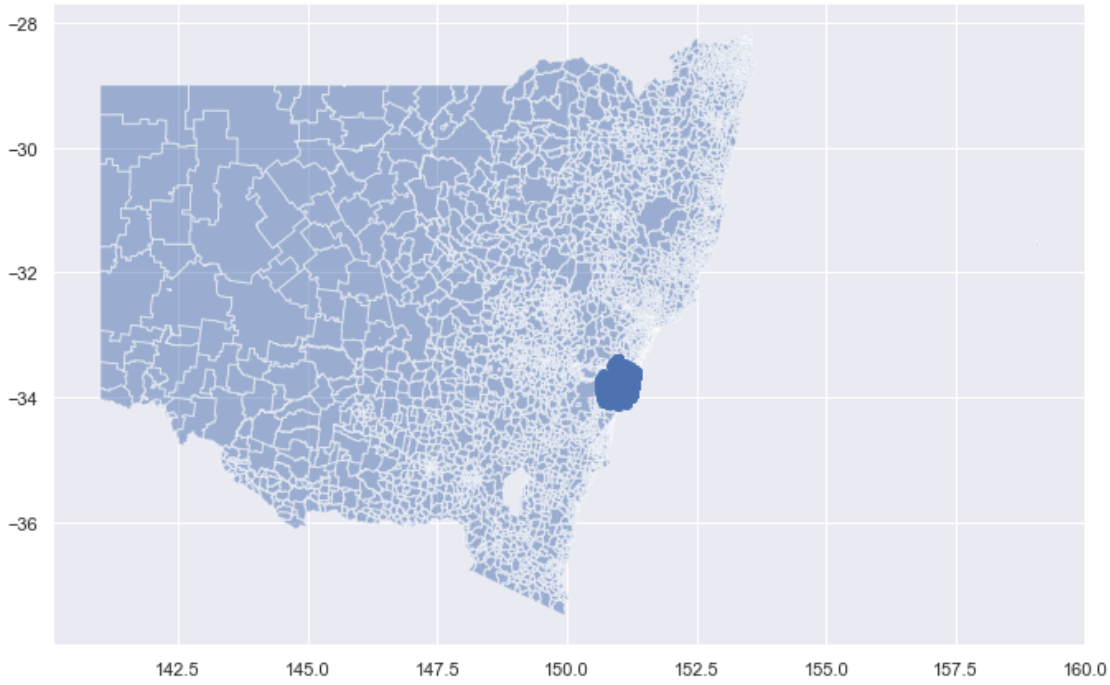
[16]:
```
df['geometry'] = [Point(x, y) for x, y in zip(df.longitude, df.latitude)]
```

[17]:
```
gdf = gpd.GeoDataFrame(df, geometry='geometry')
gdf = gdf.set_crs(epsg=4326)
```

Check that data and geospatial attributes are correct w.r.t. coordinate system used

[18]:
```
fig, ax = plt.subplots(figsize=(10, 10))
districts.plot(ax=ax, alpha=0.5)
gdf.plot(ax=ax)
```

[18]: <AxesSubplot:>



[19]:
```
px.scatter_mapbox(gdf, lat='latitude', lon='longitude',␣
↪color='neighbourhood_cleansed')
```

[20]:
```
gdf.to_postgis('airbnb', engine, if_exists='replace')
```

3 Check data & test that spatial queries work

[21]:
```
df = gpd.read_postgis("SELECT * FROM districts WHERE nsw_loca_2 =␣
 ↪'CHIPPENDALE'", engine, geom_col='geometry')
df
```

[21]:
```
   lc_ply_pid   dt_create dt_retire loc_pid  nsw_locali nsw_loca_1  \
0       24239  2015-02-24       None  NSW921  2015-05-08       None

     nsw_loca_2 nsw_loca_3 nsw_loca_4 nsw_loca_5 nsw_loca_6 nsw_loca_7  \
0  CHIPPENDALE       None       None          G       None          1

                                              geometry
0  POLYGON ((151.19334 -33.88784, 151.19317 -33.8…
```

[22]:
```
!psql -d test -c "SELECT ST_Area(geometry, true) FROM districts WHERE␣
 ↪nsw_loca_2 = 'NEWTOWN';"
```

```
      st_area
-------------------
 1586153.340185988
(1 row)
```

[23]:
```
cursor = engine.execute("SELECT ST_Area(geometry, true) FROM districts WHERE␣
 ↪nsw_loca_2 = 'NEWTOWN';")
cursor.fetchall()
```

[23]:
```
[(1586153.340185988,)]
```

[24]:
```
sql = """
SELECT roads.*
FROM roads, districts
WHERE districts.nsw_loca_2 = 'CHIPPENDALE'
    AND ST_Intersects(districts.geometry, roads.geometry);
"""
```

[25]:
```
%%time
roads = gpd.read_postgis(sql, engine, geom_col='geometry')
roads
```

```
CPU times: user 33 ms, sys: 5.54 ms, total: 38.6 ms
Wall time: 74.4 ms
```

[25]:
```
         osm_id  code       fclass                     name   ref oneway  \
0     46940608  5122  residential              Rose Street  None      B
1    173486845  5141      service            McAlister Lane  None      B
```

```
2    183111044  5141      service                        None  None      B
3    318667959  5155        steps                        None  None      B
4    411017203  5122  residential                        None  None      B
..         …   …           …                          …    …    …
301  514985726  5153      footway                        None  None      B
302  625461200  5141      service  Sydney Yard Access Bridge  None      B
303  652573453  5153      footway                        None  None      B
304    1954876  5113      primary           Cleveland Street  None      F
305  172425691  5112        trunk                  City Road   A36      F

     maxspeed  layer bridge tunnel  \
0          40      0      F      F
1          40      0      F      F
2           0      0      F      F
3           0      0      F      F
4           0      0      F      F
..        …    …    …    …
301         0      0      F      F
302        20      1      T      F
303         0      0      F      F
304        50      0      F      F
305        60      0      F      F

                                              geometry
0    LINESTRING (151.19436 -33.88800, 151.19439 -33…
1    LINESTRING (151.19918 -33.88739, 151.19919 -33…
2    LINESTRING (151.20648 -33.88545, 151.20641 -33…
3    LINESTRING (151.20054 -33.88544, 151.20054 -33…
4    LINESTRING (151.19738 -33.88444, 151.19738 -33…
..                                                 …
301  LINESTRING (151.19872 -33.88741, 151.19872 -33…
302  LINESTRING (151.20196 -33.88753, 151.20211 -33…
303  LINESTRING (151.20152 -33.88546, 151.20181 -33…
304  LINESTRING (151.19310 -33.88761, 151.19325 -33…
305  LINESTRING (151.19325 -33.88769, 151.19318 -33…

[306 rows x 11 columns]
```

[26]:
```
df.crs
```

[26]:
```
<Geographic 2D CRS: EPSG:4326>
Name: WGS 84
Axis Info [ellipsoidal]:
- Lat[north]: Geodetic latitude (degree)
- Lon[east]: Geodetic longitude (degree)
Area of Use:
- name: World
```

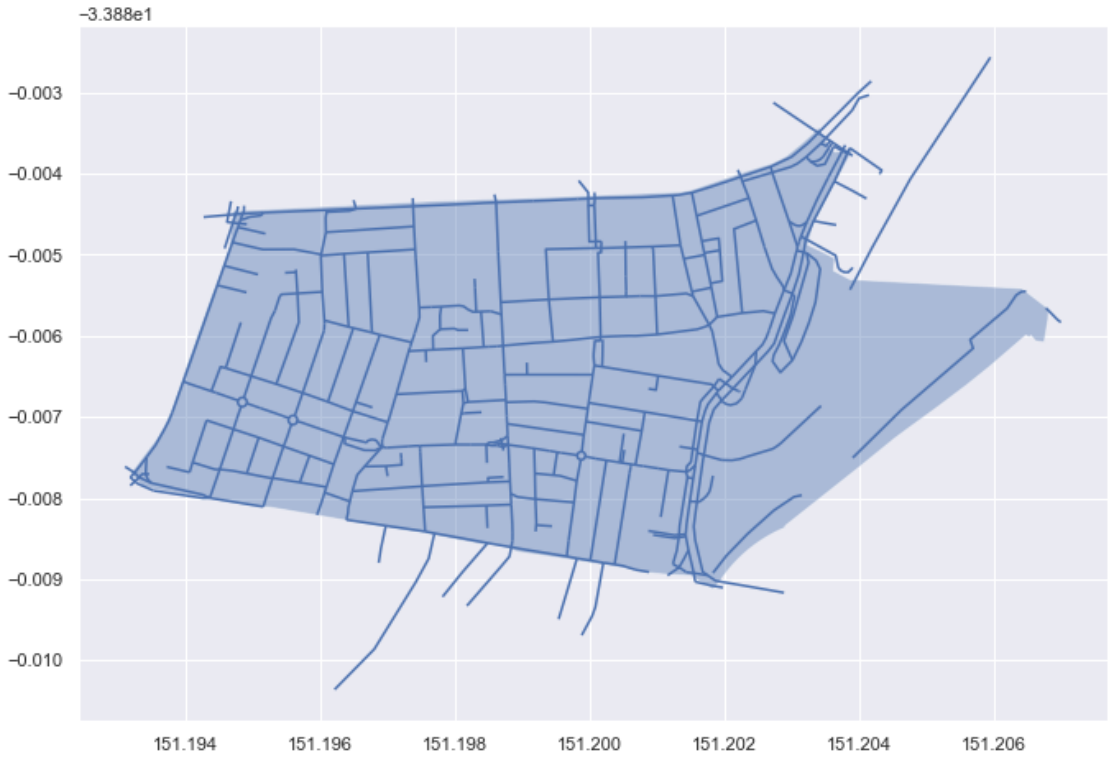
```
- bounds: (-180.0, -90.0, 180.0, 90.0)
Datum: World Geodetic System 1984
- Ellipsoid: WGS 84
- Prime Meridian: Greenwich
```

[27]:
```
roads.crs
```

[27]:
```
<Geographic 2D CRS: EPSG:4326>
Name: WGS 84
Axis Info [ellipsoidal]:
- Lat[north]: Geodetic latitude (degree)
- Lon[east]: Geodetic longitude (degree)
Area of Use:
- name: World
- bounds: (-180.0, -90.0, 180.0, 90.0)
Datum: World Geodetic System 1984
- Ellipsoid: WGS 84
- Prime Meridian: Greenwich
```

[28]:
```
fig, ax = plt.subplots(figsize=(10, 10))
ax = df.plot(alpha=0.4, ax=ax)
roads.plot(ax=ax)
```

[28]:
```
<AxesSubplot:>
```



[29]:
```
cursor = engine.execute("SELECT COUNT(*) FROM roads")
cursor.fetchall()
```

[29]:
```
[(1905150,)]
```

[30]:
```
cursor = engine.execute("SELECT COUNT(*) FROM districts")
cursor.fetchall()
```

[30]:
```
[(4591,)]
```

[31]:
```
%%time
sql = """
SELECT districts.nsw_loca_2 AS area, roads.*
FROM districts
JOIN roads ON ST_Intersects(districts.geometry, roads.geometry);
"""

roads = gpd.read_postgis(sql, engine, geom_col='geometry')
roads
```

```
CPU times: user 13 s, sys: 1.69 s, total: 14.7 s
Wall time: 2min 12s
```

[31]:

	area	osm_id	code	fclass	name	ref	oneway	\
0	CAMPSIE	1881386	5122	residential	Anglo Road	None	F	
1	CAMPSIE	1881388	5122	residential	Loch Street	None	B	
2	CAMPSIE	1881389	5115	tertiary	Evaline Street	None	B	
3	GYMEA	1881520	5115	tertiary	Clements Parade	None	B	
4	KIRRAWEE	1881520	5115	tertiary	Clements Parade	None	B	
…	…	…	…	…	…	…	…	
518483	KATOOMBA	865498461	5153	footway	None	None	B	
518484	KATOOMBA	865498462	5153	footway	None	None	B	
518485	KATOOMBA	865498463	5153	footway	None	None	B	
518486	KATOOMBA	865498464	5153	footway	None	None	B	
518487	KATOOMBA	865498465	5153	footway	None	None	B	

	maxspeed	layer	bridge	tunnel	\
0	50	0	F	F	
1	50	0	F	F	
2	50	0	F	F	
3	0	0	F	F	
4	0	0	F	F	
…	…	…	…	…	
518483	0	0	F	F	
518484	0	0	F	F	
518485	0	0	F	F	

```
518486         0      0      F      F
518487         0      0      F      F

                                                  geometry
0       LINESTRING (151.09789 -33.91319, 151.09910 -33…
1       LINESTRING (151.09770 -33.91285, 151.09775 -33…
2       LINESTRING (151.09782 -33.91558, 151.09783 -33…
3       LINESTRING (151.08040 -34.03458, 151.08030 -34…
4       LINESTRING (151.08040 -34.03458, 151.08030 -34…
…                                                     …
518483  LINESTRING (150.31060 -33.71498, 150.31061 -33…
518484  LINESTRING (150.31061 -33.71491, 150.31063 -33…
518485  LINESTRING (150.31069 -33.71476, 150.31068 -33…
518486  LINESTRING (150.31068 -33.71463, 150.31066 -33…
518487  LINESTRING (150.31066 -33.71461, 150.31075 -33…

[518488 rows x 12 columns]
```

3.1 Test query: filter roads / airbnb data

[32]:
```
%%time
sql = """
SELECT *
FROM districts
WHERE nsw_loca_2 = 'CHIPPENDALE'
"""

sydney = gpd.read_postgis(sql, engine, geom_col='geometry')
sydney
```

```
CPU times: user 22 ms, sys: 1.84 ms, total: 23.9 ms
Wall time: 30.4 ms
```

[32]:
```
   lc_ply_pid   dt_create dt_retire loc_pid  nsw_locali nsw_loca_1  \
0       24239  2015-02-24      None  NSW921  2015-05-08       None

     nsw_loca_2 nsw_loca_3 nsw_loca_4 nsw_loca_5 nsw_loca_6 nsw_loca_7  \
0  CHIPPENDALE       None       None          G       None          1

                                             geometry
0  POLYGON ((151.19334 -33.88784, 151.19317 -33.8…
```

[33]:
```
%%time
sql = """SELECT * FROM airbnb"""
listings = gpd.read_postgis(sql, engine, geom_col='geometry')
listings
```

```
CPU times: user 2.29 s, sys: 233 ms, total: 2.52 s
Wall time: 3.29 s
```

[33]:

	id	listing_url	scrape_id	last_scraped	name
0	12351	https://www.airbnb.com/rooms/12351	20181207034750	2018-12-07	Sydney City & Harbour at the door
1	14250	https://www.airbnb.com/rooms/14250	20181207034750	2018-12-07	Manly Harbour House
2	15253	https://www.airbnb.com/rooms/15253	20181207034750	2018-12-07	Stunning Penthouse Apartment In Heart Of The City
3	20865	https://www.airbnb.com/rooms/20865	20181207034750	2018-12-07	3 BED HOUSE + 1 BED STUDIO Balmain
4	26174	https://www.airbnb.com/rooms/26174	20181207034750	2018-12-07	COZY PRIVATE ROOM, GREAT LOCATION!
…	…	…	…	…	…
36657	30592081	https://www.airbnb.com/rooms/30592081	20181207034750	2018-12-07	The top floor paradise
36658	30592161	https://www.airbnb.com/rooms/30592161	20181207034750	2018-12-07	Sydney harbour catamaran sailing. Parties for 10
36659	30592248	https://www.airbnb.com/rooms/30592248	20181207034750	2018-12-07	Double Room built-in a large and bright apartment
36660	30592505	https://www.airbnb.com/rooms/30592505	20181207034750	2018-12-07	Bright Modern apartment in a Premiere location
36661	30593866	https://www.airbnb.com/rooms/30593866	20181207034750	2018-12-07	Huge Sunny Double Room with Spa in Best Location

	summary	space
0	Come stay with Vinh & Stuart (Awarded as one o…	We're pretty relaxed hosts, and we fully appre…
1	Beautifully renovated, spacious and quiet, our…	Our home is a thirty minute walk along the sea…
2	Penthouse living in a great central location: …	A charming two-level, two-bedroom, two-bathroo…
3	Hi! We are a married professional couple with …	HOUSE : _____ * DUCTED AIR CONDITIONING IN…
4	None	
…	…	
36657	This top floor paradise offers a very bright a…	
36658	Organise your function/party for 10 people on …	
36659	New Double Room with light filled interiors id…	
36660	Generous space with 2 bed rooms ,2 bathrooms ,…	
36661	Beautiful natural sunny apartment with spa, po…	

```
4      Double bed in decent sized bedroom, in two bed…
…                                                    …
36657                                               None
36658  The catamaran is spacious and ideal for functi…
36659  8 min from Homebush train Station 3 min from B…
36660  2 bed rooms both with a Queen size bed . one w…
36661  Let's talk about the room. This room is fully …

                                             description experiences_offered  \
0      Come stay with Vinh & Stuart (Awarded as one o…                  none
1      Beautifully renovated, spacious and quiet, our…                  none
2      Penthouse living in a great central location: …                  none
3      Hi! We are a married professional couple with …                  none
4      Double bed in decent sized bedroom, in two bed…                  none
…                                                    …                   …
36657  This top floor paradise offers a very bright a…                  none
36658  Organise your function/party for 10 people on …                  none
36659  New Double Room with light filled interiors id…                  none
36660  Generous space with 2 bed rooms ,2 bathrooms ,…                  none
36661  Beautiful natural sunny apartment with spa, po…                  none

                                  neighborhood_overview  … license  \
0      Pyrmont is an inner-city village of Sydney, on…  …    None
1      Balgowlah Heights is one of the most prestigio…  …    None
2      The location is really central and there is nu…  …    None
3      BALMAIN is an older inner city village / subur…  …    None
4                                                   None  …    None
…                                                    …  …     …
36657                                               None  …    None
36658  Experience Sydney harbour in a spectacular and…  …    None
36659                                               None  …    None
36660  3-5 Min Walk for Train Station / Cafe/ Restaur…  …    None
36661                                               None  …    None

      jurisdiction_names instant_bookable is_business_travel_ready  \
0                   None                f                         f
1                   None                f                         f
2                   None                t                         f
3                   None                f                         f
4                   None                f                         f
…                    …                  …                         …
36657               None                t                         f
36658               None                f                         f
36659               None                t                         f
36660               None                t                         f
36661               None                t                         f
```

```
               cancellation_policy require_guest_profile_picture  \
0      strict_14_with_grace_period                             t
1      strict_14_with_grace_period                             f
2      strict_14_with_grace_period                             f
3      strict_14_with_grace_period                             t
4                         moderate                             f
…                            …                           …
36657                     flexible                             f
36658                     flexible                             f
36659                     flexible                             f
36660  strict_14_with_grace_period                             f
36661                     flexible                             f

      require_guest_phone_verification calculated_host_listings_count  \
0                                    t                               2
1                                    f                               2
2                                    f                               2
3                                    t                               1
4                                    f                               1
…                                  …                           …
36657                                f                               1
36658                                f                               1
36659                                f                               1
36660                                f                               1
36661                                f                               3

      reviews_per_month                    geometry
0                  4.83  POINT (151.19190 -33.86515)
1                  0.03  POINT (151.26172 -33.80093)
2                  3.63  POINT (151.21654 -33.88045)
3                  0.18  POINT (151.17275 -33.85907)
4                  0.45  POINT (151.25940 -33.88909)
…                 …                          …
36657               NaN  POINT (151.25938 -33.87494)
36658               NaN  POINT (151.14957 -33.84784)
36659               NaN  POINT (151.08172 -33.86372)
36660               NaN  POINT (151.15066 -33.92996)
36661               NaN  POINT (151.21126 -33.90694)

[36662 rows x 97 columns]
```

[34]:
```
listings.to_crs(epsg=3857, inplace=True)
roads.to_crs(epsg=3857, inplace=True)
sydney.to_crs(epsg=3857, inplace=True)
sydney.crs
```

[34]:
```
<Projected CRS: EPSG:3857>
Name: WGS 84 / Pseudo-Mercator
Axis Info [cartesian]:
- X[east]: Easting (metre)
- Y[north]: Northing (metre)
Area of Use:
- name: World - 85°S to 85°N
- bounds: (-180.0, -85.06, 180.0, 85.06)
Coordinate Operation:
- name: Popular Visualisation Pseudo-Mercator
- method: Popular Visualisation Pseudo Mercator
Datum: World Geodetic System 1984
- Ellipsoid: WGS 84
- Prime Meridian: Greenwich
```

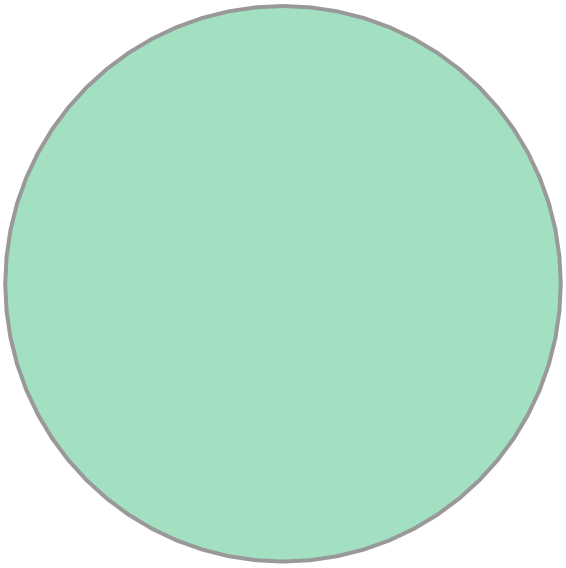
[35]:
```
sydney.geometry[0].area
```

[35]:
```
674791.8687212318
```

10km radius

[36]:
```
sydney['geometry'] = sydney.geometry[0].centroid.buffer(10000)
sydney['geometry'][0]
```

[36]:



Filter roads to Sydney (approximate)

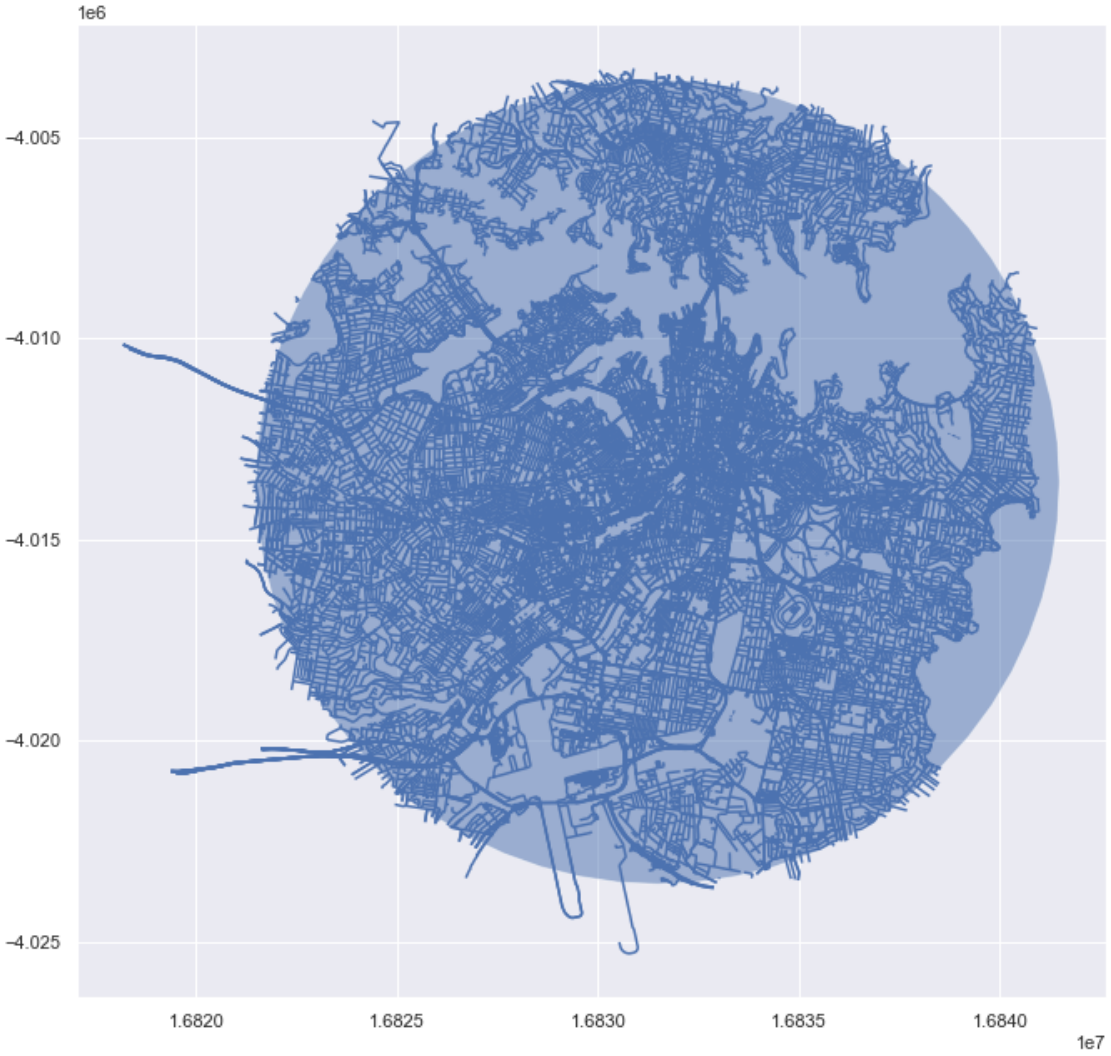
[37]:
```
print(roads.shape)
sydney_roads = roads[roads['geometry'].intersects(sydney['geometry'][0])].copy()
```

```
print(sydney_roads.shape)
```

```
(518488, 12)
(36714, 12)
```

[38]:
```
fig, ax = plt.subplots(figsize=(10, 10))
sydney.plot(ax=ax, alpha=0.5)
sydney_roads.plot(ax=ax)
```

[38]: <AxesSubplot:>



[39]:
```
crs = sydney.crs
sydney_roads.crs = crs
```

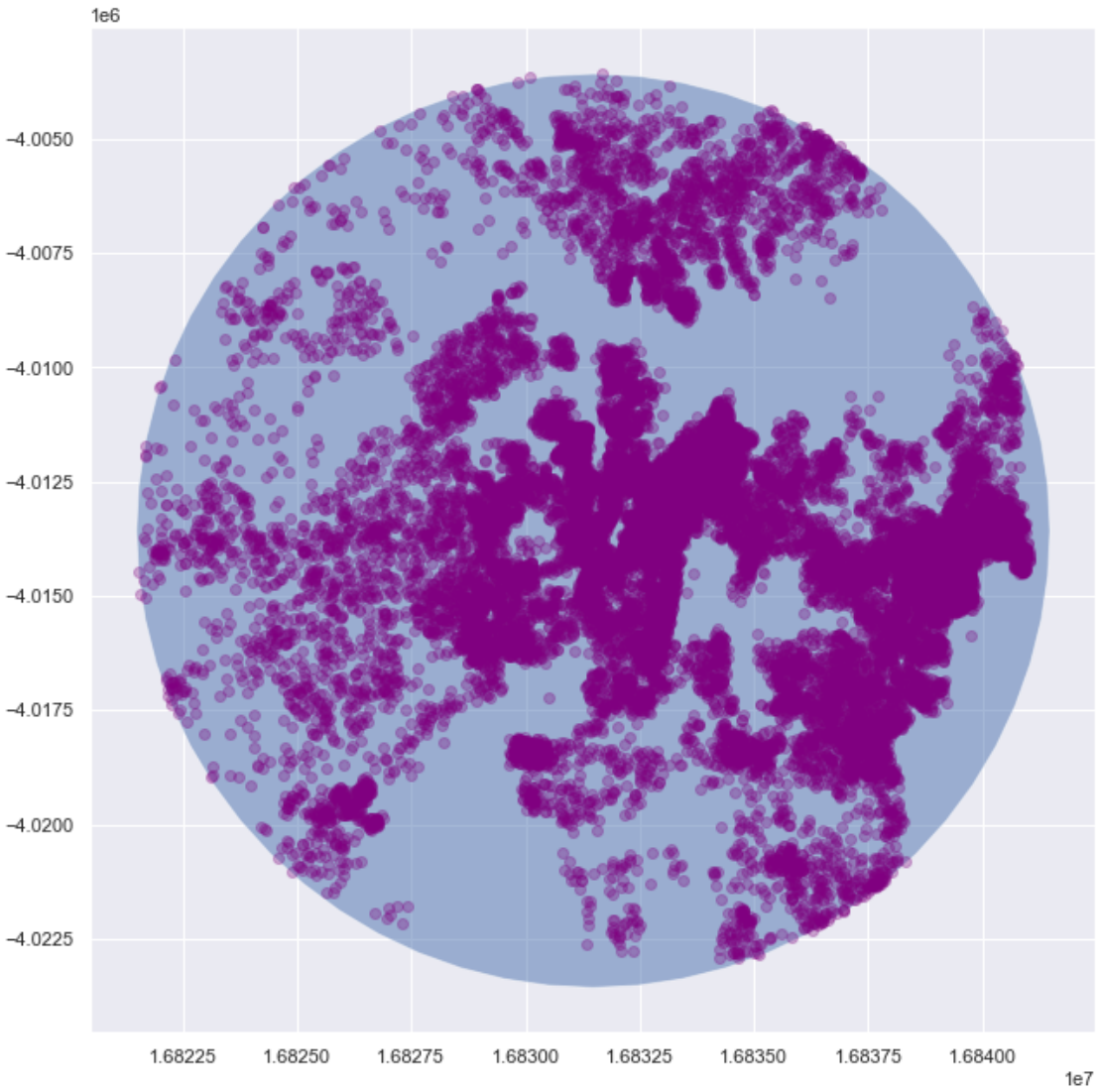
Filter Airbnb listings to Sydney (approximate)

[40]:
```
print(listings.shape)
sydney_listings = listings[listings['geometry'].
 ↪intersects(sydney['geometry'][0])].copy()
print(sydney_listings.shape)
```

```
(36662, 97)
(25088, 97)
```

[41]:
```
fig, ax = plt.subplots(figsize=(10, 10))
sydney.plot(ax=ax, alpha=0.5)
sydney_listings.plot(ax=ax, alpha=0.3, color='purple')
```

[41]: <AxesSubplot:>

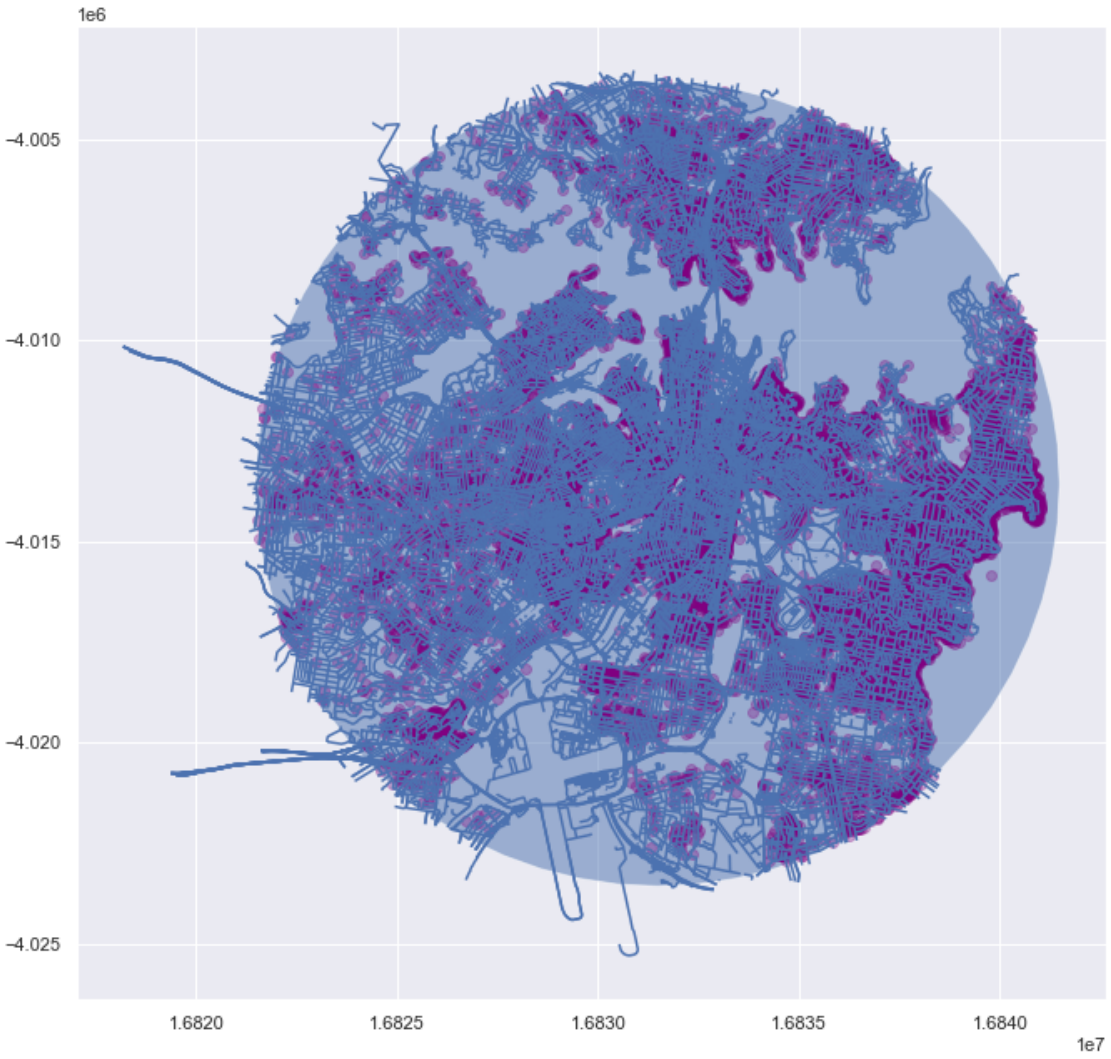


[42]:
```
crs = sydney.crs
sydney_listings.crs = crs
```

make sure all coordinate systems are consistent

[43]:
```
fig, ax = plt.subplots(figsize=(10, 10))
sydney.plot(ax=ax, alpha=0.5)
sydney_roads.plot(ax=ax)
sydney_listings.plot(ax=ax, alpha=0.3, color='purple')
```

[43]: <AxesSubplot:>



save

[44]:
```
sydney_roads.to_crs(epsg=4326, inplace=True)
sydney_roads.drop('area', 1).to_postgis('roads_sydney', engine,␣
 ↪if_exists='replace')
```

[45]:
```
sydney_listings.to_crs(epsg=4326, inplace=True)
sydney_listings.to_postgis('airbnb_sydney', engine, if_exists='replace')
```

undo

[46]:
```
engine.execute("DROP TABLE roads_sydney;")
engine.execute("DROP TABLE airbnb_sydney;")
```

[46]:
```
<sqlalchemy.engine.result.ResultProxy at 0x7fc8dc37f390>
```

4 Create test data and load to PostGIS, and save as geojson

[47]:
```
def save_gdf_as_geojson(df, path):
    js = df.to_json()
    js = geojson.loads(js)

    with open(path, 'w') as f:
        geojson.dump(js, f)
```

4.1 Create test roads data

[48]:
```
%%time
sql = """
SELECT districts.nsw_loca_2 AS area, roads.*
FROM districts
JOIN roads ON ST_Intersects(districts.geometry, roads.geometry);
"""

roads = gpd.read_postgis(sql, engine, geom_col='geometry')
print(roads.shape)
roads.head()
```

```
(518488, 12)
CPU times: user 12.9 s, sys: 891 ms, total: 13.8 s
Wall time: 26.1 s
```

[48]:

	area	osm_id	code	fclass	name	ref	oneway	\
0	MAYFIELD WEST	574897136	5112	trunk	Maitland Road	A43	F	
1	MAYFIELD WEST	603148932	5141	service	None	None	B	
2	MAYFIELD WEST	676662522	5113	primary	Tourle Street	B63	F	
3	MAYFIELD WEST	710230358	5113	primary	Tourle Street	B63	F	
4	MAYFIELD WEST	710230371	5113	primary	Maitland Road	None	F	

```
   maxspeed  layer bridge tunnel  \
0        80      0      F      F
1         0      0      F      F
2        80      0      F      F
3        60      0      F      F
4        60      0      F      F

                                                geometry
0  LINESTRING (151.71879 -32.88202, 151.71875 -32…
1  LINESTRING (151.72985 -32.88293, 151.73005 -32…
2  LINESTRING (151.73265 -32.88195, 151.73270 -32…
3  LINESTRING (151.73209 -32.88605, 151.73212 -32…
4  LINESTRING (151.72049 -32.88666, 151.72030 -32…
```

518k roads in NSW

[49]:
```
roads.to_postgis('sydney_roads', engine, if_exists='replace')
```

[50]:
```
test_set_sizes = [100, 1000, 10000, 100000, 500000]

for n in test_set_sizes:
    n_str = str(n)

    if n >= 1000000:
        n_str = n_str[:-6] + 'm'
    elif n > 100:
        n_str = n_str[:-3] + 'k'

    table_name = 'roads_sydney_' + n_str
    sampled_df = roads.sample(n)
    sampled_df.to_postgis(table_name, engine, if_exists='replace')
    save_gdf_as_geojson(sampled_df, f'data/{table_name}.geojson')

    print(f'roads_sydney_{n_str} exported')
```

```
roads_sydney_100 exported
roads_sydney_1k exported
roads_sydney_10k exported
roads_sydney_100k exported
roads_sydney_500k exported
```

4.2 Create test listings data

[51]:
```
%%time
sql = """
SELECT districts.nsw_loca_2 AS area, airbnb.*
FROM districts
```

```
JOIN airbnb ON ST_Within(airbnb.geometry, districts.geometry);
"""

listings = gpd.read_postgis(sql, engine, geom_col='geometry')
print(listings.shape)
listings.head()
```

```
(36646, 98)
CPU times: user 2.1 s, sys: 287 ms, total: 2.38 s
Wall time: 3.87 s
```

[51]:

	area	id	listing_url	scrape_id	last_scraped	name	summary	space	description	experiences_offered	…	license	jurisdiction_names	instant_bookable	is_business_travel_ready	\
0	PYRMONT	12351	https://www.airbnb.com/rooms/12351	20181207034750	2018-12-07	Sydney City & Harbour at the door	Come stay with Vinh & Stuart (Awarded as one o…	We're pretty relaxed hosts, and we fully appre…	Come stay with Vinh & Stuart (Awarded as one o…	none	…					
1	BALGOWLAH	14250	https://www.airbnb.com/rooms/14250	20181207034750	2018-12-07	Manly Harbour House	Beautifully renovated, spacious and quiet, our…	Our home is a thirty minute walk along the sea…	Beautifully renovated, spacious and quiet, our…	none	…					
2	DARLINGHURST	15253	https://www.airbnb.com/rooms/15253	20181207034750	2018-12-07	Stunning Penthouse Apartment In Heart Of The City	Penthouse living in a great central location: …	A charming two-level, two-bedroom, two-bathroo…	Penthouse living in a great central location: …	none	…					
3	BALMAIN	20865	https://www.airbnb.com/rooms/20865	20181207034750	2018-12-07	3 BED HOUSE + 1 BED STUDIO Balmain	Hi! We are a married professional couple with …	HOUSE : _____ * DUCTED AIR CONDITIONING IN…	Hi! We are a married professional couple with …	none	…					
4	BELLEVUE HILL	26174	https://www.airbnb.com/rooms/26174	20181207034750	2018-12-07	COZY PRIVATE ROOM, GREAT LOCATION!	None	Double bed in decent sized bedroom, in two bed…	Double bed in decent sized bedroom, in two bed…	none	…					

```
0    None                None                f                            f
1    None                None                f                            f
2    None                None                t                            f
3    None                None                f                            f
4    None                None                f                            f

           cancellation_policy require_guest_profile_picture  \
0  strict_14_with_grace_period                             t
1  strict_14_with_grace_period                             f
2  strict_14_with_grace_period                             f
3  strict_14_with_grace_period                             t
4                     moderate                             f

  require_guest_phone_verification calculated_host_listings_count  \
0                                t                               2
1                                f                               2
2                                f                               2
3                                t                               1
4                                f                               1

  reviews_per_month                    geometry
0              4.83  POINT (151.19190 -33.86515)
1              0.03  POINT (151.26172 -33.80093)
2              3.63  POINT (151.21654 -33.88045)
3              0.18  POINT (151.17275 -33.85907)
4              0.45  POINT (151.25940 -33.88909)

[5 rows x 98 columns]
```

[52]:
```
listings[['latitude', 'longitude']].describe()
```

[52]:
```
           latitude     longitude
count  36646.000000  36646.000000
mean     -33.863124    151.204240
std        0.071509      0.083528
min      -34.135212    150.642903
25%      -33.898448    151.179670
50%      -33.882491    151.215903
75%      -33.832122    151.261385
max      -33.389728    151.339811
```

[53]:
```
listings_1m = listings.copy()
n = 1000000
while len(listings_1m) < n:
    print(len(listings_1m))
    listings2 = listings_1m.copy()
```

```
    listings2.latitude = listings2.latitude + np.random.normal(0, 0.001,␣
↪size=len(listings2))
    listings2.longitude = listings2.longitude + np.random.normal(0, 0.001,␣
↪size=len(listings2))
    listings2['geometry'] = [Point(x, y) for x, y in zip(listings2.longitude,␣
↪listings2.latitude)]
    listings_1m = pd.concat([listings_1m, listings2])

listings_1m.shape
```

```
36646
73292
146584
293168
586336
```

[53]:
```
(1172672, 98)
```

[54]:
```
cols = ['id', 'listing_url', 'name', 'room_type', 'property_type',
        'summary', 'description', 'host_id',  'host_name', 'host_since',
        'street', 'neighbourhood_cleansed', 'area',
        'price', 'number_of_reviews','review_scores_rating',
        'latitude', 'longitude', 'geometry']
listings_1m = listings_1m[cols]
```

[55]:
```
listings_1m.reset_index(drop=True, inplace=True)
```

[56]:
```
test_set_sizes = [100, 1000, 10000, 100000, 500000]

for n in test_set_sizes:
    n_str = str(n)

    if n >= 1000000:
        n_str = n_str[:-6] + 'm'
    elif n > 100:
        n_str = n_str[:-3] + 'k'

    table_name = 'airbnb_sydney_' + n_str
    sampled_df = listings_1m.sample(n)
    sampled_df.to_postgis(table_name, engine, if_exists='replace')
    save_gdf_as_geojson(sampled_df, f'data/{table_name}.geojson')

    print(f'airbnb_sydney_{n_str} exported')
```

```
airbnb_sydney_100 exported
airbnb_sydney_1k exported
airbnb_sydney_10k exported
```

```
airbnb_sydney_100k exported
airbnb_sydney_500k exported
```

4.3 Check results

[57]:
```
!psql -U Kai -d test -c "\d+"
```

```
                        List of relations
 Schema |        Name         | Type  | Owner |   Size    | Description
--------+---------------------+-------+-------+-----------+-------------
 public | airbnb              | table | Kai   | 153 MB    |
 public | airbnb_sydney_100   | table | Kai   | 176 kB    |
 public | airbnb_sydney_100k  | table | Kai   | 145 MB    |
 public | airbnb_sydney_10k   | table | Kai   | 14 MB     |
 public | airbnb_sydney_1k    | table | Kai   | 1504 kB   |
 public | airbnb_sydney_1m    | table | Kai   | 1445 MB   |
 public | airbnb_sydney_500k  | table | Kai   | 722 MB    |
 public | buildings           | table | Kai   | 353 MB    |
 public | districts           | table | Kai   | 92 MB     |
 public | geography_columns   | view  | Kai   | 0 bytes   |
 public | geometry_columns    | view  | Kai   | 0 bytes   |
 public | landuse             | table | Kai   | 215 MB    |
 public | natural             | table | Kai   | 4088 kB   |
 public | places              | table | Kai   | 2928 kB   |
 public | pofw                | table | Kai   | 1192 kB   |
 public | pois                | table | Kai   | 48 MB     |
 public | railways            | table | Kai   | 10016 kB  |
 public | roads               | table | Kai   | 636 MB    |
 public | roads_sydney_100    | table | Kai   | 48 kB     |
 public | roads_sydney_100k   | table | Kai   | 39 MB     |
 public | roads_sydney_10k    | table | Kai   | 3992 kB   |
 public | roads_sydney_1k     | table | Kai   | 416 kB    |
 public | roads_sydney_500k   | table | Kai   | 192 MB    |
 public | spatial_ref_sys     | table | Kai   | 6976 kB   |
 public | sydney_roads        | table | Kai   | 200 MB    |
 public | traffic             | table | Kai   | 23 MB     |
 public | transport           | table | Kai   | 144 kB    |
 public | water               | table | Kai   | 254 MB    |
 public | waterways           | table | Kai   | 178 MB    |
(29 rows)
```

[58]:
```
!ls -lh 'data' | grep geojson
```

```
-rw-r--r--@  1 Kai  staff    69M Nov  3 17:20 airbnb_sydney.geojson
-rw-r--r--@  1 Kai  staff   171K Nov 14 15:28 airbnb_sydney_100.geojson
-rw-r--r--@  1 Kai  staff   166M Nov 14 15:29 airbnb_sydney_100k.geojson
-rw-r--r--@  1 Kai  staff    17M Nov 14 15:28 airbnb_sydney_10k.geojson
```

```
-rw-r--r--@  1 Kai  staff   1.7M Nov 14 15:28 airbnb_sydney_1k.geojson
-rw-r--r--@  1 Kai  staff   1.6G Nov  8 23:49 airbnb_sydney_1m.geojson
-rw-r--r--@  1 Kai  staff   832M Nov 14 15:31 airbnb_sydney_500k.geojson
-rw-r--r--@  1 Kai  staff   653K Nov  3 17:20 districts.geojson
-rw-r--r--@  1 Kai  staff    49K Nov 10 15:59 nsw_single_shape.geojson
-rw-r--r--@  1 Kai  staff   7.6M Nov  3 17:20 roads_sydney.geojson
-rw-r--r--@  1 Kai  staff    67K Nov 14 15:25 roads_sydney_100.geojson
-rw-r--r--@  1 Kai  staff    64M Nov 14 15:25 roads_sydney_100k.geojson
-rw-r--r--@  1 Kai  staff   6.4M Nov 14 15:25 roads_sydney_10k.geojson
-rw-r--r--@  1 Kai  staff   657K Nov 14 15:25 roads_sydney_1k.geojson
-rw-r--r--@  1 Kai  staff   637M Nov  8 23:43 roads_sydney_1m.geojson
-rw-r--r--@  1 Kai  staff   318M Nov 14 15:28 roads_sydney_500k.geojson
```

5 MongoDB

5.1 Insert data

[59]:
```python
def bulk_insert(db: Database, collection_name: str, file_path: str):
    if collection_name in db.list_collection_names():
        db.drop_collection(collection_name)

    db.create_collection(collection_name)
    collection = db[collection_name]

    # create 2dsphere index
    collection.create_index([("geometry", GEOSPHERE)])

    assert os.path.exists(file_path), 'file does not exist'

    with open(file_path, 'r') as f:
        js = geojson.load(f)

    n_features = len(js['features'])

    for feature in js['features']:
        collection.insert_one(feature)

    assert collection.count_documents({}) == n_features, 'import failed'
```

[60]:
```python
client = MongoClient('mongodb://localhost:27017/')
```

[61]:
```python
if 'test' in client.list_database_names():
    client.drop_database('test')
```

[62]:
```python
client.list_database_names()
```

[62]:
```
['a1', 'admin', 'config', 'local']
```

[63]:
```
db = client['test']
```

[64]:
```
assert len(db.list_collection_names()) == 0
```

[65]:
```
data = 'airbnb'
for n in ['100', '1k', '10k', '100k', '1m']:
    name = f'{data}_sydney_{n}'
    bulk_insert(db, name, f'data/{name}.geojson')
    print(name, 'loaded')

data = 'roads'
for n in ['100', '1k', '10k', '100k', '500k']:
    name = f'{data}_sydney_{n}'
    bulk_insert(db, name, f'data/{name}.geojson')
    print(name, 'loaded')
```

```
airbnb_sydney_100 loaded
airbnb_sydney_1k loaded
airbnb_sydney_10k loaded
airbnb_sydney_100k loaded
airbnb_sydney_1m loaded
roads_sydney_100 loaded
roads_sydney_1k loaded
roads_sydney_10k loaded
roads_sydney_100k loaded
roads_sydney_500k loaded
```

[66]:
```
bulk_insert(db, 'districts', 'data/districts.geojson')
```

[67]:
```
db.list_collection_names()
```

[67]:
```
['roads_sydney_1k',
 'roads_sydney_100k',
 'airbnb_sydney_10k',
 'roads_sydney_10k',
 'airbnb_sydney_100k',
 'airbnb_sydney_1k',
 'roads_sydney_500k',
 'roads_sydney_100',
 'airbnb_sydney_1m',
 'airbnb_sydney_100',
 'districts']
```

5.2 Test Geospatial operators

5.2.1 $nearSphere

[68]:
```
sample = db.airbnb_sydney_1m.find_one({})
print(sample['geometry'])
```

```
{'type': 'Point', 'coordinates': [151.093007, -33.765708]}
```

[69]:
```
# should return the same airbnb listing
result = db.airbnb_sydney_1m.find({"geometry": {"$nearSphere":␣
↪sample['geometry']['coordinates']}}).limit(1)
result = [item for item in result]
result[0]['_id'] == sample['_id']
```

[69]: True

$nearSphere + $maxDistance, $centerSphere

https://www.latlong.net/place/sydney-opera-house-australia-3894.html

[70]:
```
def distance_to_radians(km):
    """ convert distance (in km) to radians as required by all spherical␣
↪spatial operators in Mongo """
    return km / 6378.137

# opera house coordinates
lat = -33.856159
lon = 151.215256
point = [lon, lat]
max_dist = 5 # in km
```

[71]:
```
db.airbnb_sydney_1m.count_documents({})
```

[71]: 1000000

[72]:
```
result = db.airbnb_sydney_1m.find(
    {
        "geometry": SON([
            ("$nearSphere", point),
            ("$maxDistance", distance_to_radians(km=max_dist))
        ])
    }
)
results = [item for item in result]
len(results)
```

[72]: 319233

[73]:
```
result = db.airbnb_sydney_10k.find(
    {
        "geometry": {
            '$geoWithin': {
                '$centerSphere': [point, distance_to_radians(km=max_dist)]
            }
        }
    }
)
results = [item for item in result]
len(results)
```

[73]: 3190

[74]:
```
data = {}
data['room_type'] = [item['properties']['room_type'] for item in results]
data['property_type'] = [item['properties']['property_type'] for item in␣
 ↪results]
data['price'] = [item['properties']['price'] for item in results]
data['lat'] = [item['properties']['latitude'] for item in results]
data['lon'] = [item['properties']['longitude'] for item in results]
data = pd.DataFrame(data)
data['geometry'] = [Point(x, y) for x, y in zip(data.lon, data.lat)]
data = gpd.GeoDataFrame(data, geometry='geometry')
data = data.set_crs(epsg=4326)
fig = px.scatter_mapbox(data, lat='lat', lon='lon', height=800, width=1000,␣
 ↪zoom=12, color='property_type', opacity=0.8)
fig.show()
```

[75]:
```
sample = pd.DataFrame()
sample['lat'] = [lat]
sample['lon'] = [lon]
sample['geometry'] = [Point(x, y) for x, y in zip(sample.lon, sample.lat)]
sample = gpd.GeoDataFrame(sample, geometry='geometry')
sample = sample.set_crs(epsg=4326)
sample = sample.to_crs(epsg=3857)
data = data.to_crs(epsg=3857)
data['distance_to_opera_house_in_meters'] = data.geometry.
 ↪distance(sample['geometry'][0])
data['price'] = data['price'].str.replace('[$,]', '').astype(float)
data.head()
```

[75]:

	room_type	property_type	price	lat	lon	\
0	Entire home/apt	Apartment	120.0	-33.818895	151.238634	
1	Entire home/apt	House	143.0	-33.818606	151.237539	
2	Entire home/apt	House	630.0	-33.819132	151.235862	
3	Entire home/apt	House	775.0	-33.819357	151.234016	

```
4     Private room         House   90.0 -33.821718  151.236393

                             geometry  distance_to_opera_house_in_meters
0  POINT (16835807.741 -4004509.832)                          5631.563987
1  POINT (16835685.843 -4004471.237)                          5610.872498
2  POINT (16835499.148 -4004541.685)                          5466.845714
3  POINT (16835293.613 -4004571.754)                          5356.135969
4  POINT (16835558.280 -4004888.176)                          5180.968624
```

5.2.2 $geoIntersects / $geoWithin

Note

Spatial join is not possible in mongo db. In other words, using `$geoWithin` or `$geoIntersects` as part of `$lookup` pipeline in aggregation is not possible.

`$expr` is needed to access the variable that stores `geometry` of the subject that is performing the `$lookup`, but `$geoIntersects` or `$geoWithin` are not aggregation operators. Only `$geoNear` can be used in aggregation pipeline.

geoWithin

[76]:
```
district = db.districts.find_one({'properties.area': 'CHIPPENDALE'})
district.keys()
```

[76]:
```
dict_keys(['_id', 'type', 'id', 'geometry', 'properties'])
```

[77]:
```
result = db.airbnb_sydney_1m.find(
    {
        'geometry': {
            '$geoWithin': {'$geometry': district['geometry']}
        }
    }
)
result = [item for item in result]
len(result)
```

[77]:
```
8687
```

geoIntersects

[78]:
```
district = db.districts.find_one({'properties.area': 'CHIPPENDALE'})
district.keys()
```

[78]:
```
dict_keys(['_id', 'type', 'id', 'geometry', 'properties'])
```

[79]:
```
result = db.roads_sydney_500k.find(
    {
        'geometry': {
```

```
            '$geoIntersects': {'$geometry': district['geometry']}
        }
    }
)
result = [item for item in result]
len(result)
```

[79]: 344

6 Performance Comparison

What's being tested

PostGIS	MongoDB
ST_Within()	$geoWithin
ST_Intersects()	$geoIntersects

Operations

- Within: Find all airbnb listings in town x
- Intersects: Find all roads that intersect with town x

Note: The discrepancies in the results between PostGIS and Mongo is due to rounding. PostGIS uses 5 decimal places for lon/lat while MongoDB uses 6.

[3]:
```
client = MongoClient('mongodb://localhost:27017/')
db = client['test']
db.list_collection_names()
```

[3]:
```
['roads_sydney_1k',
 'roads_sydney_100k',
 'airbnb_sydney_10k',
 'roads_sydney_10k',
 'airbnb_sydney_100k',
 'airbnb_sydney_1k',
 'roads_sydney_500k',
 'roads_sydney_100',
 'airbnb_sydney_1m',
 'airbnb_sydney_100',
 'districts']
```

6.1 Select a polygon to use as a constant

[4]:
```
district = gpd.read_file('data/Igismap/Australia_admin_6.shp')
district = district[['name', 'place', 'way_area', 'geometry']]
district.head()
```

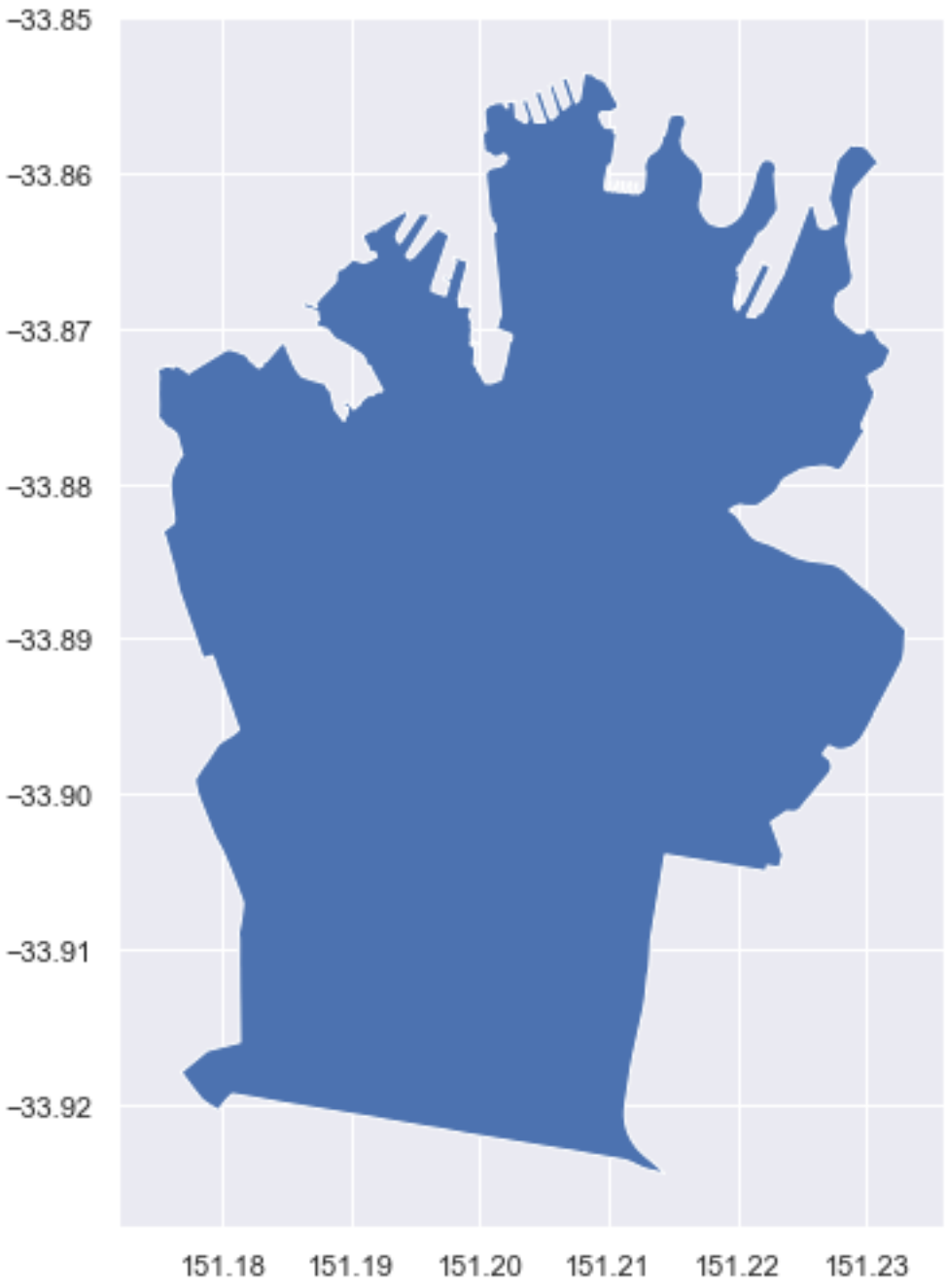
[4]:
```
                          name         place     way_area  \
0            Inner West Council  municipality   51166700.0
1  Council of the City of Sydney  municipality   38710600.0
2              Burwood Council          None   10397300.0
3   Canterbury-Bankstown Council  municipality  161810000.0
4              City of Banyule  municipality  100174000.0

                                            geometry
0  POLYGON ((151.11230 -33.89565, 151.11239 -33.8…
1  POLYGON ((151.17475 -33.87266, 151.17507 -33.8…
2  POLYGON ((151.08898 -33.89977, 151.08921 -33.8…
3  POLYGON ((150.96607 -33.94262, 150.96645 -33.9…
4  POLYGON ((145.02783 -37.76382, 145.02792 -37.7…
```

[5]:
```
district = district[district['name'] == 'Council of the City of Sydney']
district
```

[5]:
```
                          name         place    way_area  \
1  Council of the City of Sydney  municipality  38710600.0

                                            geometry
1  POLYGON ((151.17475 -33.87266, 151.17507 -33.8…
```

[6]:
```
district.plot()
```

[6]:
```
<AxesSubplot:>
```



[7]:
```
district_polygon = district['geometry'].values[0]
district_polygon
```

[7]:

[8]:
```
sql = f"""
SELECT *
FROM districts
WHERE ST_Intersects(districts.geometry, ST_GeometryFromText('{district_polygon.
 ↪wkt}', 4326));
"""

districts = gpd.read_postgis(sql, engine, geom_col='geometry')
print(districts.shape)
districts.head()
```

```
(39, 13)
```

[8]:

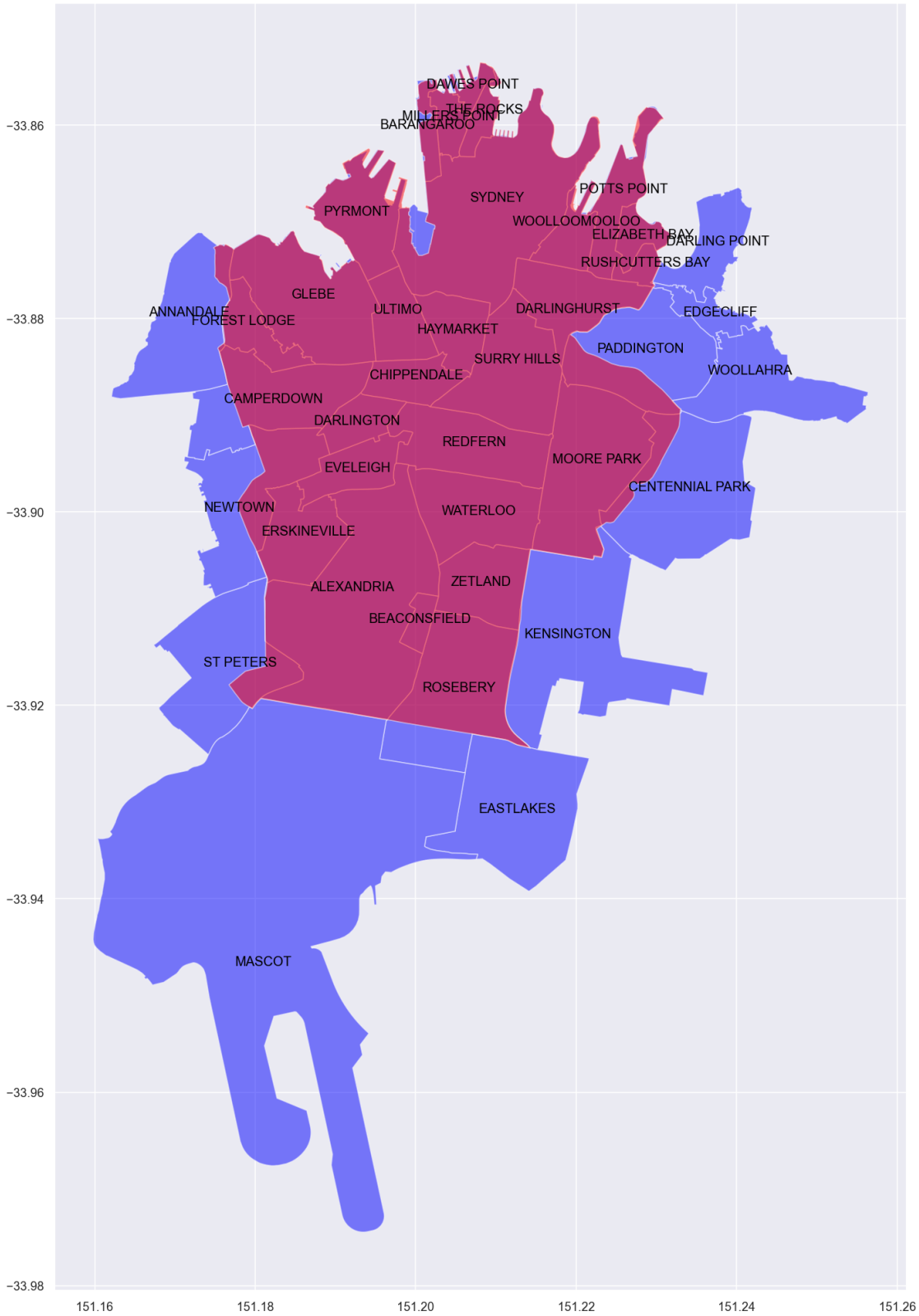
	lc_ply_pid	dt_create	dt_retire	loc_pid	nsw_locali	nsw_loca_1	\
0	24908	2015-09-03	None	NSW887	2015-11-10	None	
1	14765	2011-05-17	None	NSW4555	2008-09-11	None	
2	29986	2019-05-29	None	NSW1412	2017-05-02	None	
3	29987	2019-05-29	None	NSW3282	2012-10-30	None	
4	26621	2016-09-12	None	NSW3475	2016-11-11	None	

	nsw_loca_2	nsw_loca_3	nsw_loca_4	nsw_loca_5	nsw_loca_6	nsw_loca_7	\
0	CENTENNIAL PARK	None	None	G	None	1	
1	ZETLAND	None	None	G	None	1	
2	ELIZABETH BAY	None	None	G	None	1	
3	POTTS POINT	None	None	G	None	1	
4	RUSHCUTTERS BAY	None	None	G	None	1	

	geometry
0	POLYGON ((151.23414 -33.88984, 151.23524 -33.8…
1	POLYGON ((151.20953 -33.90370, 151.20968 -33.9…
2	POLYGON ((151.22752 -33.86809, 151.22751 -33.8…
3	POLYGON ((151.22987 -33.85812, 151.22991 -33.8…
4	POLYGON ((151.22892 -33.87227, 151.22900 -33.8…

[9]:
```
districts['coords'] = districts['geometry'].apply(lambda x: x.
 ↪representative_point().coords[:])
districts['coords'] = [coords[0] for coords in districts['coords']]
```

[10]:
```
fig, ax = plt.subplots(figsize=(20, 20), dpi=120)
ax = districts.plot(ax=ax, alpha=0.5, color='blue')
ax = district.plot(alpha=0.5, ax=ax, color='red')

for idx, row in districts.iterrows():
    plt.annotate(text=row['nsw_loca_2'], xy=row['coords'],␣
 ↪horizontalalignment='center', color='black')
```


DAWES POINT
THE ROCKS
MILLERS POINT
BARANGAROO
SYDNEY
POTTS POINT
PYRMONT
WOOLLOOMOOLOO
ELIZABETH BAY
DARLING POINT
RUSHCUTTERS BAY
GLEBE
ANNANDALE
FOREST LODGE
ULTIMO
DARLINGHURST
EDGECLIFF
HAYMARKET
PADDINGTON
SURRY HILLS
WOOLLAHRA
CHIPPENDALE
CAMPERDOWN
DARLINGTON
REDFERN
MOORE PARK
EVELEIGH
CENTENNIAL PARK
NEWTOWN
WATERLOO
ERSKINEVILLE
ALEXANDRIA
ZETLAND
BEACONSFIELD
KENSINGTON
ST PETERS
ROSEBERY
EASTLAKES
MASCOT
−33.86
−33.88
−33.90
−33.92
−33.94
−33.96
−33.98
151.16
151.18
151.20
151.22
151.24
151.26

[11]:
```
district_polygon = district['geometry'].values[0]
district_polygon
```

[11]:

[12]:
```
js = district.to_json()
js = geojson.loads(js)
```

6.2 Query 1: Intersection

[13]:
```
test_set_sizes = [100, 1000, 10000, 100000, 500000]
```

6.2.1 PostGIS

`ST_Intersects()`

```
SELECT *
FROM roads
WHERE ST_Intersects(roads.geometry, ST_GeometryFromText(text WKT, SRID));
```

Example: `ST_GeomFromText('POINT(-126.4 45.32)', 312)`

[14]:
```
%%time
sql = f"""
SELECT *
FROM roads as roads
WHERE ST_Intersects(roads.geometry, ST_GeometryFromText('{district_polygon.
 ↪wkt}', 4326));
"""
df = gpd.read_postgis(sql, engine, geom_col='geometry')
print(df.shape)
df.head()
```

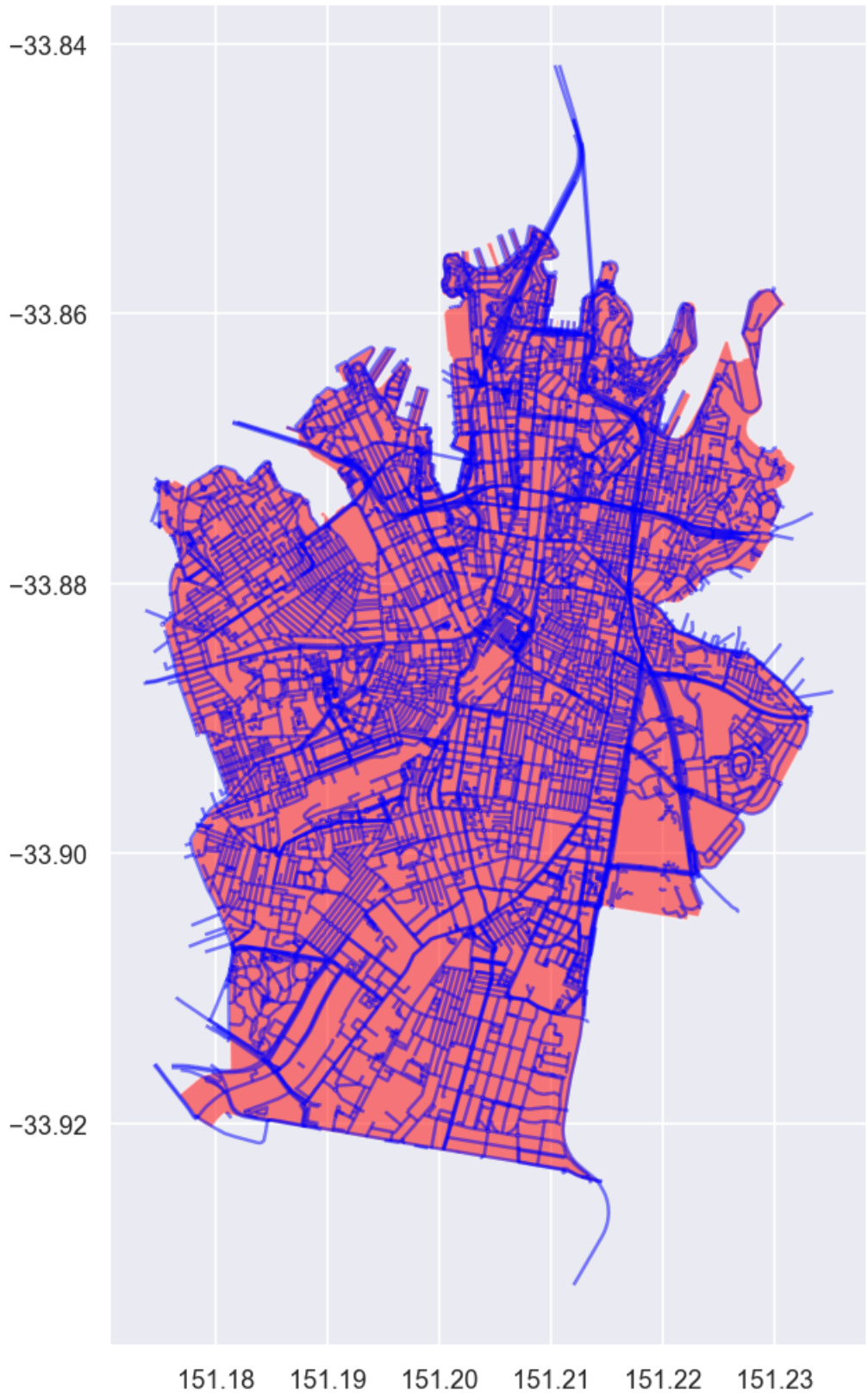
```
(10201, 11)
CPU times: user 216 ms, sys: 12.2 ms, total: 228 ms
Wall time: 482 ms
```

[14]:

	osm_id	code	fclass	name	ref	oneway	maxspeed	layer	\
0	1884057	5114	secondary	Park Street	None	B	40	0	
1	1954876	5113	primary	Cleveland Street	None	F	50	0	
2	2077098	5153	footway	None	None	B	0	0	

```
3  1954879  5115   tertiary              None  None      B         0      0
4  1955001  5115   tertiary         Lang Road  None      B        50      0

  bridge tunnel                                              geometry
0      F      F  LINESTRING (151.20813 -33.87312, 151.20805 -33…
1      F      F  LINESTRING (151.19310 -33.88761, 151.19325 -33…
2      F      F  LINESTRING (151.21720 -33.86053, 151.21709 -33…
3      F      F  LINESTRING (151.23214 -33.89222, 151.23213 -33…
4      F      F  LINESTRING (151.23221 -33.89212, 151.23234 -33…
```

[15]:
```
fig, ax = plt.subplots(figsize=(10, 10), dpi=120)
ax = district.plot(alpha=0.5, ax=ax, color='red')
ax = df.plot(ax=ax, alpha=0.5, color='blue')
```


−33.84
−33.86
−33.88
−33.90
−33.92
151.18
151.19
151.20
151.21
151.22
151.23

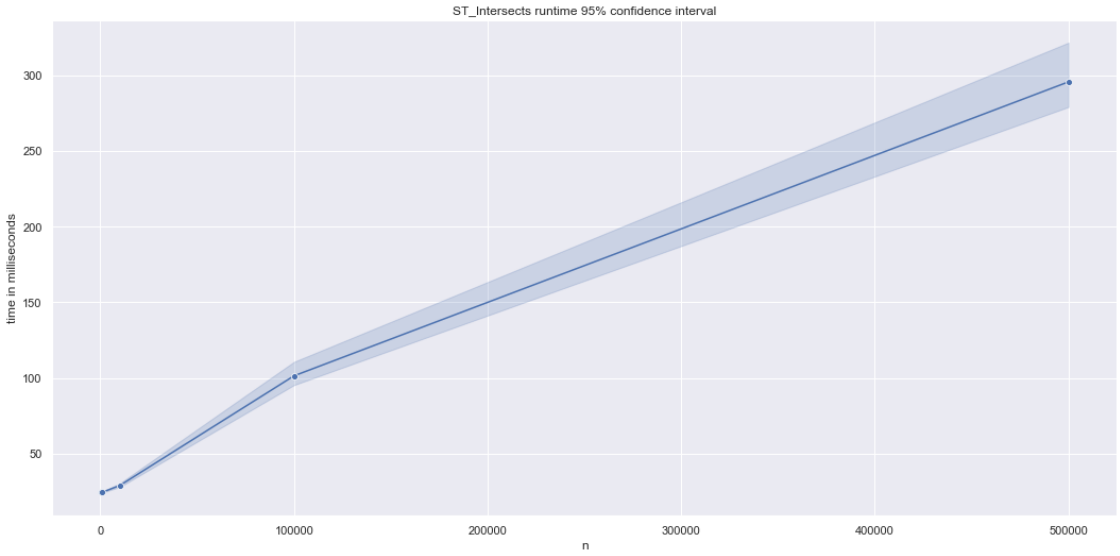
[16]:
```
data = []

for n in test_set_sizes:
    n_str = str(n)

    if n >= 1000000:
        n_str = n_str[:-6] + 'm'
    elif n > 100:
        n_str = n_str[:-3] + 'k'

    table_name = f'roads_sydney_{n_str}'

    sql = f"""
        SELECT *
        FROM {table_name} as roads
        WHERE ST_Intersects(roads.geometry,␣
 ↪ST_GeometryFromText('{district_polygon.wkt}', 4326));
    """

    for i in range(50):
        start = time()
        roads = gpd.read_postgis(sql, engine, geom_col='geometry')
        end = time()
        data += (n, (end - start) * 1000),

    print(table_name, 'done. Shape: ', roads.shape)
```

```
roads_sydney_100 done. Shape:  (4, 12)
roads_sydney_1k done. Shape:  (17, 12)
roads_sydney_10k done. Shape:  (213, 12)
roads_sydney_100k done. Shape:  (2153, 12)
roads_sydney_500k done. Shape:  (10792, 12)
```

[17]:
```
df = pd.DataFrame(data, columns=['n', 'time in milliseconds'])
```

[18]:
```
plt.title('ST_Intersects runtime 95% confidence interval')
sns.lineplot(data=df, x='n', y='time in milliseconds', marker='o')
```

[18]:
```
<AxesSubplot:title={'center':'ST_Intersects runtime 95% confidence interval'},
xlabel='n', ylabel='time in milliseconds'>
```



[19]:
```
all_runtime = df.copy()
all_runtime['system'] = 'PostGIS'
```

6.2.2 MongoDB

[20]:
```
%%time
cursor = db['roads_sydney_500k'].find(
    {
        'geometry': {
            '$geoIntersects': {'$geometry': js['features'][0]['geometry']}
        }
    }
)
print(len([item for item in cursor]), 'documents')
stats = cursor.explain()
time_in_ms = stats['executionStats']['executionTimeMillis']
print(time_in_ms, 'ms')
```

```
10781 documents
3868 ms
CPU times: user 167 ms, sys: 17 ms, total: 184 ms
Wall time: 11.9 s
```

[21]:
```
stats['executionStats']['executionTimeMillis']
```

[21]: 3868

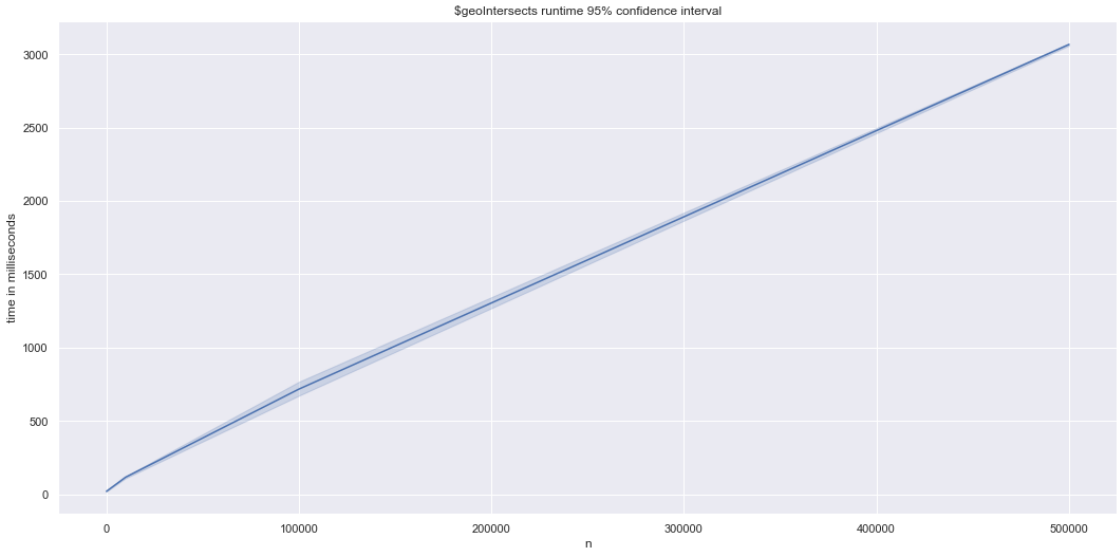
[22]:
```
data = []

for n in test_set_sizes:
    n_str = str(n)

    if n >= 1000000:
        n_str = n_str[:-6] + 'm'
    elif n > 100:
        n_str = n_str[:-3] + 'k'

    table_name = f'roads_sydney_{n_str}'
    logic = {
        'geometry': {
            '$geoIntersects': {
                '$geometry': js['features'][0]['geometry']
            }
        }
    }

    for i in range(50):
        cursor = db[table_name].find(logic)
        stats = cursor.explain()
        time_in_ms = stats['executionStats']['executionTimeMillis']
        data += (n, time_in_ms),

    print(table_name, 'done. Matched records: ', len([item for item in cursor]))
```

```
roads_sydney_100 done. Matched records:  4
roads_sydney_1k done. Matched records:  17
roads_sydney_10k done. Matched records:  213
roads_sydney_100k done. Matched records:  2145
roads_sydney_500k done. Matched records:  10781
```

[23]:
```
df = pd.DataFrame(data, columns=['n', 'time in milliseconds'])
df['system'] = 'MongoDB'
```

[24]:
```
plt.title('$geoIntersects runtime 95% confidence interval')
sns.lineplot(data=df, x='n', y='time in milliseconds', markers='o')
```

[24]:
```
<AxesSubplot:title={'center':'$geoIntersects runtime 95% confidence interval'},
xlabel='n', ylabel='time in milliseconds'>
```

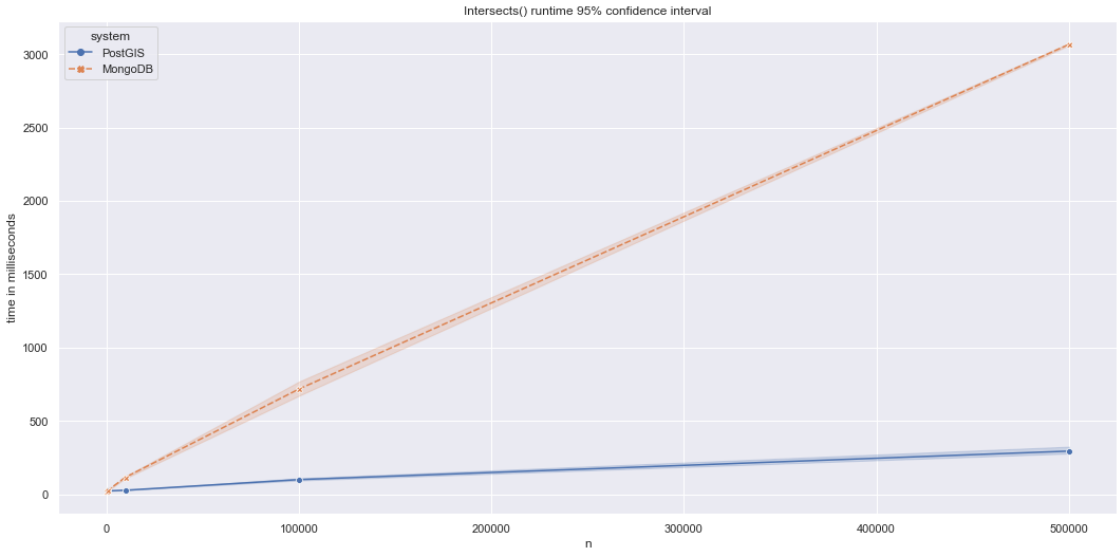


6.2.3 Query 1 Result

[25]:
```
all_runtime = pd.concat([all_runtime, df])
```

[26]:
```
plt.title('Intersects() runtime 95% confidence interval')
sns.lineplot(data=all_runtime, x='n', y='time in milliseconds', hue='system',␣
↪style='system', markers=True)
```

[26]:
```
<AxesSubplot:title={'center':'Intersects() runtime 95% confidence interval'},
xlabel='n', ylabel='time in milliseconds'>
```



[27]:
```
grouped = all_runtime.groupby(['system', 'n']).agg(['mean', 'std'])
grouped = grouped.unstack(0)
grouped = grouped.iloc[:, [1, 3, 0, 2]]
grouped = grouped.swaplevel(axis=1)
grouped
```

[27]:

	time in milliseconds			
system	PostGIS		MongoDB	
	mean	std	mean	std
n				
100	24.474568	2.226587	21.44	5.334371
1000	24.641771	1.129567	30.26	6.865501
10000	29.034467	4.607614	117.56	25.955346
100000	101.443100	29.581162	717.60	176.065155
500000	295.494485	87.394251	3065.40	27.723121

[28]:
```
# how to query hierarchical index columns
# grouped.loc[:, grouped.columns.get_level_values(1) == 'mean']
# grouped.loc[:, (grouped.columns.get_level_values(1) == 'mean') & (grouped.
 ↪columns.get_level_values(2) == 'MongoDB')]
```

[29]:
```
df = grouped.copy()
df.columns = ['PostGIS_mean', 'PostGIS_std', 'MongoDB_mean', 'MongoDB_std']
df
```

[29]:

	PostGIS_mean	PostGIS_std	MongoDB_mean	MongoDB_std
n				
100	24.474568	2.226587	21.44	5.334371
1000	24.641771	1.129567	30.26	6.865501
10000	29.034467	4.607614	117.56	25.955346
100000	101.443100	29.581162	717.60	176.065155
500000	295.494485	87.394251	3065.40	27.723121

[107]:
```
# prevent zero division if 0
for col in ['MongoDB_mean', 'MongoDB_std']:
    df.loc[df[col] == 0, col] = 0.0001
```

[32]:
```
df.reset_index(inplace=True)
```

[108]:
```
fig = px.line(df, x='n', y=['MongoDB_mean', 'PostGIS_mean'],
              log_x=True, log_y=True, height=600, width=1000,
              labels={'value': 'time in milliseconds', 'MongoDB_mean':␣
 ↪'MongoDB', 'PostGIS_mean': 'PostGIS'},
              title='Intersects() Runtime Comparison between PostGIS and␣
 ↪MongoDB')

for data in fig.data:
```

```
    data.update(mode='markers+lines')
fig
```

[35]:
```
fig = go.Figure()
fig.add_trace(go.Scatter(x=df['n'],
                         y=df['PostGIS_mean'],
                         mode='lines+markers',
                         name='PostGIS',
                         error_y=dict(type='data', # value of error bar given␣
 ↪in data coordinates
                                      array=df['PostGIS_std'],
                                      visible=True)))
fig.add_trace(go.Scatter(x=df['n'],
                         y=df['MongoDB_mean'],
                         mode='lines+markers',
                         name='MongoDB',
                         error_y=dict(type='data', # value of error bar given␣
 ↪in data coordinates
                                      array=df['MongoDB_std'],
                                      visible=True)))
fig.update_xaxes(type="log")
fig.update_yaxes(type="log")
fig.update_layout(title='Intersects() Runtime Comparison between PostGIS and␣
 ↪MongoDB',
                  xaxis_title='n',
                  yaxis_title='Time in milliseconds')
fig.show()
```

6.3 Query 2: Within

[36]:
```
test_set_sizes = [100, 1000, 10000, 100000, 1000000]
```

6.3.1 PostGIS

ST_Within()

```
SELECT *
FROM a
WHERE ST_Within(a.geometry, b.geometry);
-- Evaluates to True if a is inside of b
```

[37]:
```
%%time
sql = f"""
SELECT *
FROM airbnb
WHERE ST_Within(airbnb.geometry, ST_GeometryFromText('{district_polygon.wkt}',␣
 ↪4326));
```

```
"""
df = gpd.read_postgis(sql, engine, geom_col='geometry')
print(df.shape)
df.head(1)
```

```
(9243, 97)
CPU times: user 432 ms, sys: 96.8 ms, total: 528 ms
Wall time: 3.16 s
```

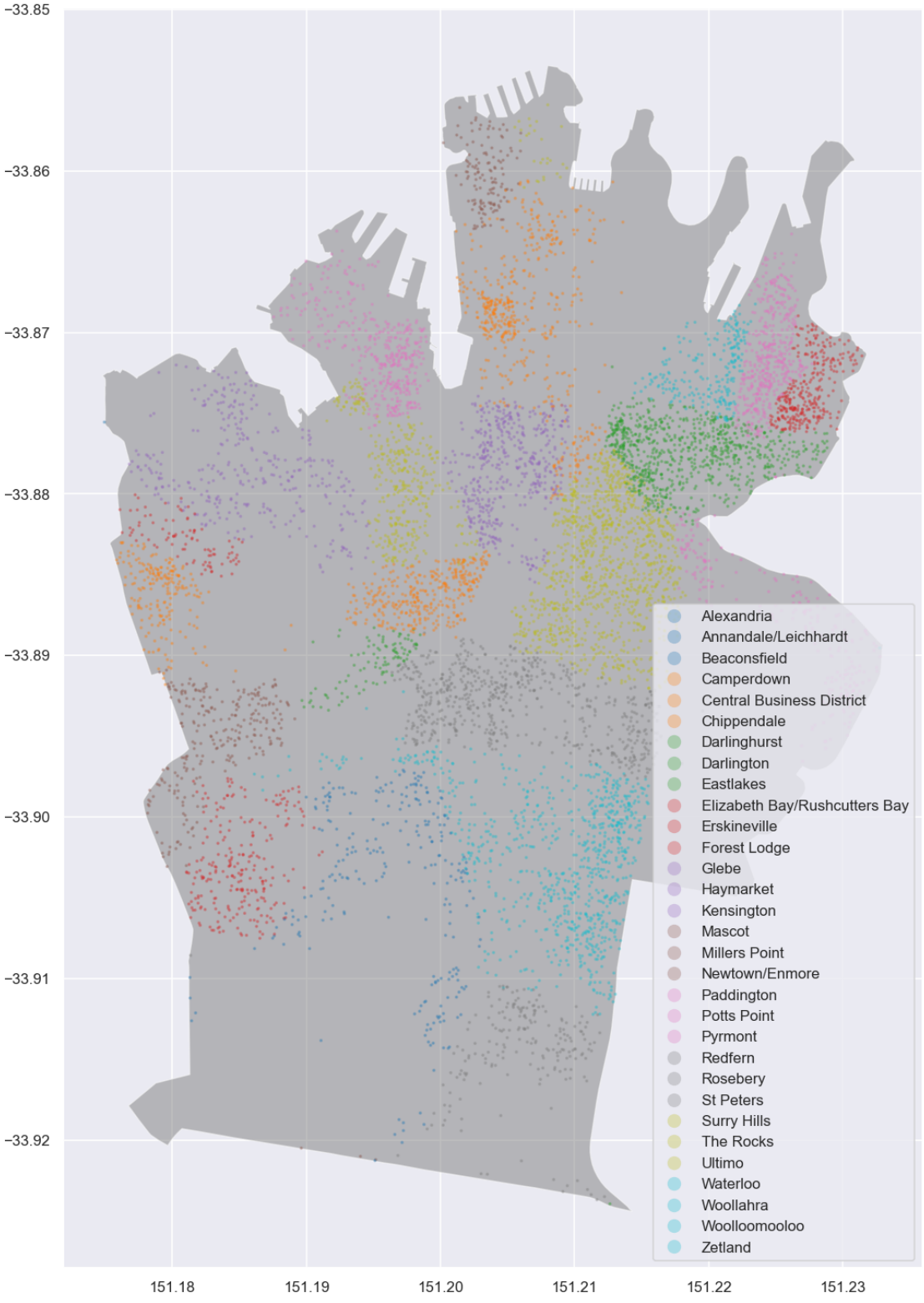
[37]:
```
        id                          listing_url       scrape_id last_scraped  \
0  310414  https://www.airbnb.com/rooms/310414  20181207034750   2018-12-07

                                          name  \
0  Relax on the Terrace of a Darlinghurst Home

                                             summary  \
0  Soak up the sunshine on a lounger at this inne…

                                               space  \
0  Welcome to Sydney! This quirky terrace house h…

                                         description experiences_offered  \
0  Soak up the sunshine on a lounger at this inne…                 none

                              neighborhood_overview  … license  \
0  Victoria Street, Stanley Street, and Crown Str…  …    None

  jurisdiction_names instant_bookable is_business_travel_ready  \
0               None                f                         f

           cancellation_policy require_guest_profile_picture  \
0  strict_14_with_grace_period                             f

  require_guest_phone_verification calculated_host_listings_count  \
0                                f                               1

  reviews_per_month                     geometry
0              1.24  POINT (151.21766 -33.88191)

[1 rows x 97 columns]
```

[38]:
```
df.iloc[:1, 20:40]
```

[38]:
```
                                  host_url host_name  host_since  \
0  https://www.airbnb.com/users/show/1596677     David  2012-01-11

                            host_location  \
```

```
0  Darlinghurst, New South Wales, Australia

                                          host_about host_response_time  \
0  Inner city Sydney guy, loves to travel, but ha…               None

  host_response_rate host_acceptance_rate host_is_superhost  \
0               None                 None                 t

                                  host_thumbnail_url  \
0  https://a0.muscache.com/im/users/1596677/profi…

                                    host_picture_url host_neighbourhood  \
0  https://a0.muscache.com/im/users/1596677/profi…         Paddington

   host_listings_count  host_total_listings_count  \
0                  1.0                       1.0

                                  host_verifications host_has_profile_pic  \
0  ['email', 'phone', 'facebook', 'reviews', 'jum…                     t

  host_identity_verified                        street neighbourhood  \
0                      t  Darlinghurst, NSW, Australia    Paddington

  neighbourhood_cleansed
0                 Sydney
```

[39]:
```
fig, ax = plt.subplots(figsize=(10, 20), dpi=110)
ax = district.plot(alpha=0.5, ax=ax, color='grey')
ax = df.plot('neighbourhood', ax=ax, alpha=0.3, markersize=2, legend=True)
```


−33.85
−33.86
−33.87
−33.88
−33.89
−33.90
−33.91
−33.92
151.18
151.19
151.20
151.21
151.22
151.23
Alexandria
Annandale/Leichhardt
Beaconsfield
Camperdown
Central Business District
Chippendale
Darlinghurst
Darlington
Eastlakes
Elizabeth Bay/Rushcutters Bay
Erskineville
Forest Lodge
Glebe
Haymarket
Kensington
Mascot
Millers Point
Newtown/Enmore
Paddington
Potts Point
Pyrmont
Redfern
Rosebery
St Peters
Surry Hills
The Rocks
Ultimo
Waterloo
Woollahra
Woolloomooloo
Zetland

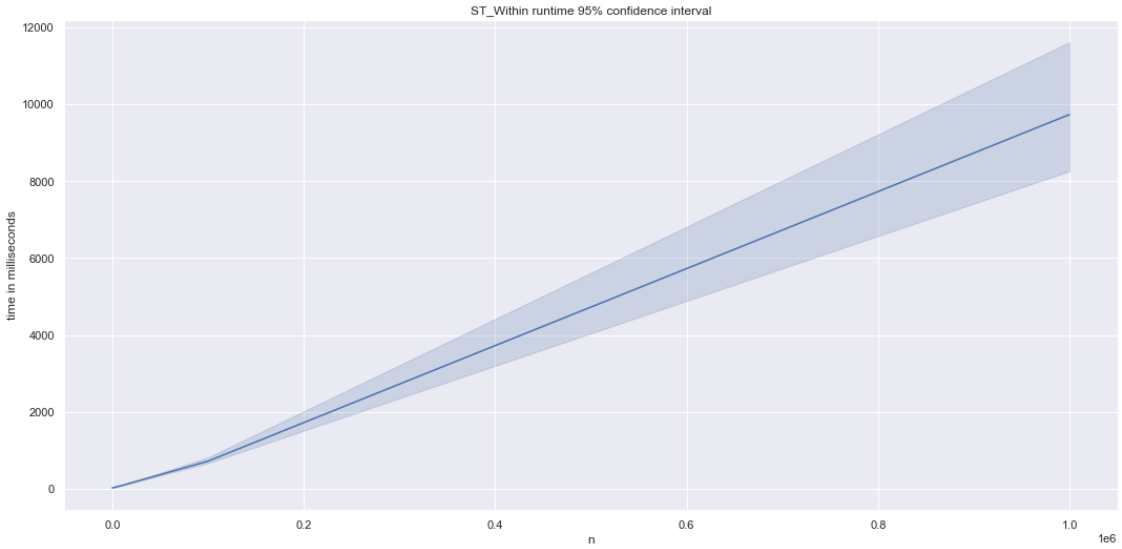
[40]:
```
data = []

for n in test_set_sizes:
    n_str = str(n)

    if n >= 1000000:
        n_str = n_str[:-6] + 'm'
    elif n > 100:
        n_str = n_str[:-3] + 'k'

    table_name = f'airbnb_sydney_{n_str}'

    sql = f"""
        SELECT *
        FROM {table_name} as airbnb
        WHERE ST_Within(airbnb.geometry, ST_GeometryFromText('{district_polygon.
 ↪wkt}', 4326));
    """

    for i in range(50):
        start = time()
        listings = gpd.read_postgis(sql, engine, geom_col='geometry')
        end = time()
        data += (n, (end - start) * 1000),

    print(table_name, 'done. Shape: ', listings.shape)
```

```
airbnb_sydney_100 done. Shape:  (32, 19)
airbnb_sydney_1k done. Shape:  (253, 19)
airbnb_sydney_10k done. Shape:  (2460, 19)
airbnb_sydney_100k done. Shape:  (24837, 19)
airbnb_sydney_1m done. Shape:  (247129, 19)
```

[41]:
```
df = pd.DataFrame(data, columns=['n', 'time in milliseconds'])
```

[42]:
```
plt.title('ST_Within runtime 95% confidence interval')
sns.lineplot(data=df, x='n', y='time in milliseconds', markers='o')
```

[42]:
```
<AxesSubplot:title={'center':'ST_Within runtime 95% confidence interval'},
xlabel='n', ylabel='time in milliseconds'>
```



[43]:
```
all_runtime = df.copy()
all_runtime['system'] = 'PostGIS'
```

6.3.2 MongoDB

[44]:
```
%%time
cursor = db['airbnb_sydney_1m'].find(
    {
        'geometry': {
            '$geoWithin': {'$geometry': js['features'][0]['geometry']}
        }
    }
)
print(len([item for item in cursor]), 'documents')
stats = cursor.explain()
time_in_ms = stats['executionStats']['executionTimeMillis']
print(time_in_ms)
```

```
247276 documents
8644
CPU times: user 3.73 s, sys: 833 ms, total: 4.56 s
Wall time: 32.5 s
```

[45]:
```
data = []

for n in test_set_sizes:
    n_str = str(n)
```

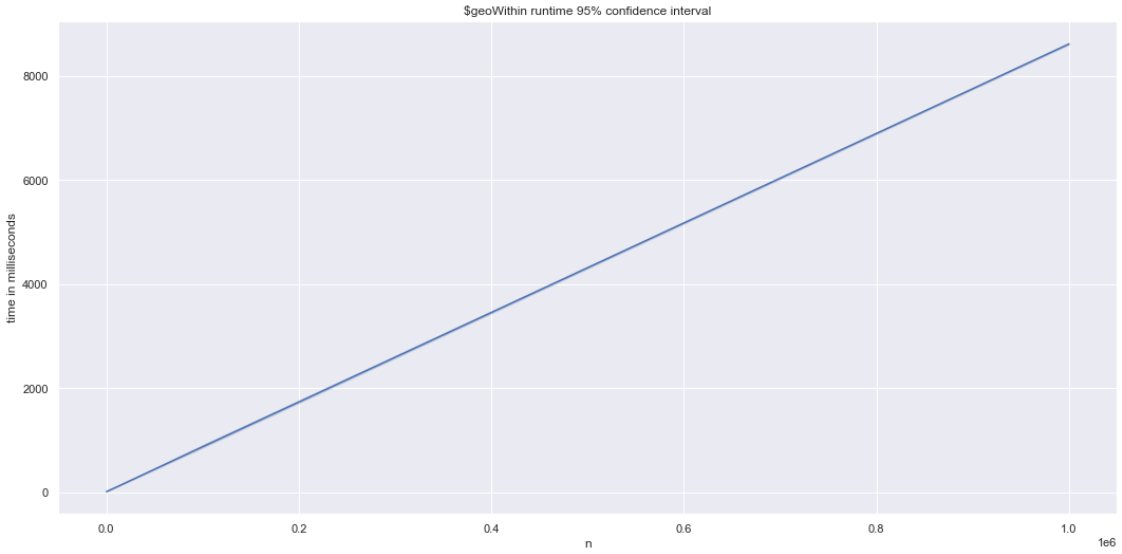
```
    if n >= 1000000:
        n_str = n_str[:-6] + 'm'
    elif n > 100:
        n_str = n_str[:-3] + 'k'

    table_name = f'airbnb_sydney_{n_str}'
    logic = {
        'geometry': {
            '$geoWithin': {
                '$geometry': js['features'][0]['geometry']
            }
        }
    }

    for i in range(50):
        cursor = db[table_name].find(logic)
        stats = cursor.explain()
        time_in_ms = stats['executionStats']['executionTimeMillis']
        data += (n, time_in_ms),

    print(table_name, 'done. Matched records: ', len([item for item in cursor]))
```

```
airbnb_sydney_100 done. Matched records:  32
airbnb_sydney_1k done. Matched records:  253
airbnb_sydney_10k done. Matched records:  2460
airbnb_sydney_100k done. Matched records:  24837
airbnb_sydney_1m done. Matched records:  247276
```

[46]:
```
df = pd.DataFrame(data, columns=['n', 'time in milliseconds'])
df['system'] = 'MongoDB'
```

[47]:
```
plt.title('$geoWithin runtime 95% confidence interval')
sns.lineplot(data=df, x='n', y='time in milliseconds', markers='o')
```

[47]:
```
<AxesSubplot:title={'center':'$geoWithin runtime 95% confidence interval'},
xlabel='n', ylabel='time in milliseconds'>
```

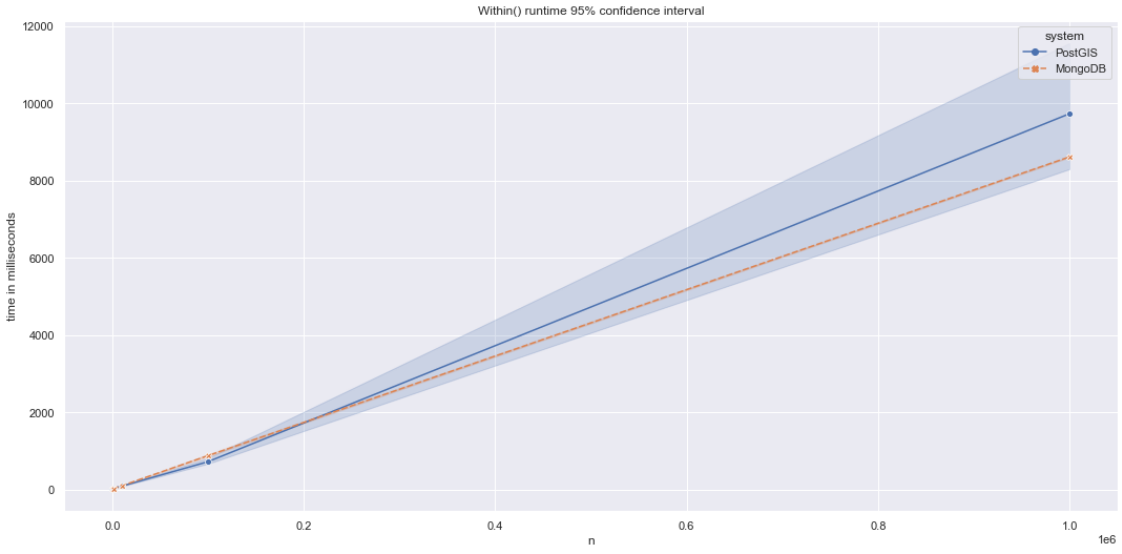


6.3.3 Query 2 Result

[48]:
```
all_runtime = pd.concat([all_runtime, df])
```

[49]:
```
plt.title('Within() runtime 95% confidence interval')
sns.lineplot(data=all_runtime, x='n', y='time in milliseconds', hue='system',␣
↪style='system', markers=True)
```

[49]:
```
<AxesSubplot:title={'center':'Within() runtime 95% confidence interval'},
xlabel='n', ylabel='time in milliseconds'>
```



[50]:
```
grouped = all_runtime.groupby(['system', 'n']).agg(['mean', 'std'])
grouped = grouped.unstack(0)
grouped = grouped.iloc[:, [1, 3, 0, 2]]
grouped = grouped.swaplevel(axis=1)
grouped
```

[50]:

	time in milliseconds			
system	PostGIS		MongoDB	
	mean	std	mean	std
n				
100	25.076060	2.370539	15.98	2.867766
1000	30.708113	2.946591	22.50	1.644409
10000	87.724705	21.697864	97.58	15.732574
100000	719.415178	322.328978	875.58	117.794025
1000000	9726.424212	6086.861840	8610.30	46.200208

[51]:
```
# how to query hierarchical index columns
# grouped.loc[:, grouped.columns.get_level_values(1) == 'mean']
# grouped.loc[:, (grouped.columns.get_level_values(1) == 'mean') & (grouped.
 ↪columns.get_level_values(2) == 'MongoDB')]
```

[52]:
```
df = grouped.copy()
df.columns = ['PostGIS_mean', 'PostGIS_std', 'MongoDB_mean', 'MongoDB_std']
df
```

[52]:

	PostGIS_mean	PostGIS_std	MongoDB_mean	MongoDB_std
n				
100	25.076060	2.370539	15.98	2.867766
1000	30.708113	2.946591	22.50	1.644409
10000	87.724705	21.697864	97.58	15.732574
100000	719.415178	322.328978	875.58	117.794025
1000000	9726.424212	6086.861840	8610.30	46.200208

[53]:
```
# prevent zero division if 0
for col in ['MongoDB_mean', 'MongoDB_std']:
    df.loc[df[col] == 0, col] = 0.0001
```

[54]:
```
df.reset_index(inplace=True)
```

[55]:
```
fig = px.line(df, x='n', y=['MongoDB_mean', 'PostGIS_mean'],
              log_x=True, log_y=True, height=600, width=1000,
              labels={'value': 'time in seconds', 'MongoDB_mean': 'MongoDB',␣
 ↪'PostGIS_mean': 'PostGIS'},
              title='Within() Runtime Comparison between PostGIS and MongoDB')

for data in fig.data:
    data.update(mode='markers+lines')
```

```
fig
```

[56]:
```
fig = go.Figure()
fig.add_trace(go.Scatter(x=df['n'],
                         y=df['PostGIS_mean'],
                         mode='lines+markers',
                         name='PostGIS',
                         error_y=dict(type='data', # value of error bar given␣
 ↪in data coordinates
                                      array=df['PostGIS_std'],
                                      visible=True)))
fig.add_trace(go.Scatter(x=df['n'],
                         y=df['MongoDB_mean'],
                         mode='lines+markers',
                         name='MongoDB',
                         error_y=dict(type='data', # value of error bar given␣
 ↪in data coordinates
                                      array=df['MongoDB_std'],
                                      visible=True)))
fig.update_xaxes(type="log")
fig.update_yaxes(type="log")
fig.update_layout(title='Within() Runtime Comparison between PostGIS and␣
 ↪MongoDB',
                  xaxis_title='n',
                  yaxis_title='Time in milliseconds')
fig.show()
```

7 ST_Relates

7.1 Cross intersection

[ ]:
```
%%time
sql = f"""
    -- convert polygon to circle where centroid is Chippendale and radius is␣
 ↪10km
    WITH geo_filter AS (
        SELECT ST_Transform(ST_Buffer(ST_Centroid(ST_Transform(geometry,␣
 ↪3857)), 10000), 4326)
        FROM districts
        WHERE nsw_loca_2 = 'CHIPPENDALE'
    ),
    roads_subset AS (
        SELECT DISTINCT *
        FROM roads
        WHERE ST_Intersects(roads.geometry, (SELECT geometry FROM geo_filter))
    )
```

```
    SELECT r1.osm_id AS r1_osm_id
        , r1.fclass AS r1_fclass
        , r1.name AS r1_name
        , r1.geometry AS r1_geometry
        , r2.osm_id AS r2_osm_id
        , r2.fclass AS r2_fclass
        , r2.name AS r2_name
        , ST_AsText(r2.geometry) AS r2_geometry
    FROM roads_subset AS r1
    JOIN roads_subset AS r2 ON ST_Intersects(r1.geometry, r2.geometry)
    WHERE ST_Relate(r1.geometry, r2.geometry, '0F1FF0102')
        AND r1.name > r2.name;
"""

df = gpd.read_postgis(sql, engine, geom_col='r1_geometry')
print(df.shape)
df.head()
```

[31]:
```
%%time
sql = f"""
    WITH roads_subset AS (
        SELECT DISTINCT *
        FROM roads
        WHERE ST_Intersects(roads.geometry,␣
 ↪ST_GeometryFromText('{district_polygon.wkt}', 4326))
    )
    SELECT r1.osm_id AS r1_osm_id
        , r1.fclass AS r1_fclass
        , r1.name AS r1_name
        , r1.geometry AS r1_geometry
        , r2.osm_id AS r2_osm_id
        , r2.fclass AS r2_fclass
        , r2.name AS r2_name
        , ST_AsText(r2.geometry) AS r2_geometry
    FROM roads_subset AS r1
    JOIN roads_subset AS r2 ON ST_Intersects(r1.geometry, r2.geometry)
    WHERE ST_Relate(r1.geometry, r2.geometry, 'F010F0102');
"""

df = gpd.read_postgis(sql, engine, geom_col='r1_geometry')
print(df.shape)
df.head()
```

```
(6, 8)
CPU times: user 29.7 ms, sys: 3.81 ms, total: 33.5 ms
Wall time: 44.6 s
```

[31]:
```
   r1_osm_id r1_fclass r1_name  \
0  143528258   footway    None
1  194616057   service    None
2  194616059   service    None
3  209795720   service    None
4  209795721   service    None

                                        r1_geometry  r2_osm_id r2_fclass  \
0  LINESTRING (151.21274 -33.87361, 151.21275 -33…    2948478   footway
1  LINESTRING (151.18819 -33.91986, 151.18909 -33…  194616059   service
2  LINESTRING (151.18904 -33.92017, 151.18816 -33…  194616057   service
3  LINESTRING (151.19407 -33.89044, 151.19422 -33…  209795721   service
4  LINESTRING (151.19418 -33.89048, 151.19412 -33…  209795720   service

  r2_name                                        r2_geometry
0    None  LINESTRING(151.2148659 -33.8701192,151.2148815…
1    None  LINESTRING(151.1890405 -33.9201711,151.1881568…
2    None  LINESTRING(151.1881929 -33.9198573,151.189086 …
3    None  LINESTRING(151.194175 -33.8904839,151.1941246 …
4    None  LINESTRING(151.194066 -33.890444,151.1942206 -…
```

[25]:
```
df['r1_osm_id'].isin(df['r2_osm_id']).value_counts()
```

[25]:
```
True    8
Name: r1_osm_id, dtype: int64
```

[26]:
```
df1 = df[[col for col in df.columns if col.startswith('r1')]].copy()
df1.rename(columns={'r1_geometry': 'geometry'}, inplace=True)
df2 = df[[col for col in df.columns if col.startswith('r2')]].copy()
df2['r2_geometry'] = df2['r2_geometry'].apply(wkt.loads)
df2.rename(columns={'r2_geometry': 'geometry'}, inplace=True)
df2 = gpd.GeoDataFrame(df2, geometry='geometry')
```

[27]:
```
map1 = KeplerGl()
map1
```

```
User Guide: https://docs.kepler.gl/docs/keplergl-jupyter

KeplerGl()
```

[28]:
```
map1.add_data(df1, name='roads1')
```

[29]:
```
map1.add_data(df2, name='roads2')
```

[19]:
```
map1.save_to_html(file_name='map1.html')
```

```
Map saved to map1.html!
```

7.2 T-intersection

[20]:
```
%%time
sql = f"""
    WITH roads AS (
        SELECT *
        FROM roads_sydney_500k AS roads
        WHERE ST_Intersects(roads.geometry,␣
 ↪ST_GeometryFromText('{district_polygon.wkt}', 4326))
    )
    SELECT r1.area AS r1_area
        , r1.osm_id AS r1_osm_id
        , r1.fclass AS r1_fclass
        , r1.name AS r1_name
        , r1.geometry AS r1_geometry
        , r2.area AS r2_area
        , r2.osm_id AS r2_osm_id
        , r2.fclass AS r2_fclass
        , r2.name AS r2_name
        , ST_AsText(r2.geometry) AS r2_geometry
    FROM roads AS r1
    JOIN roads AS r2 ON ST_Intersects(r1.geometry, r2.geometry)
    WHERE ST_Relate(r1.geometry, r2.geometry, 'F010F0102');
"""

df = gpd.read_postgis(sql, engine, geom_col='r1_geometry')
print(df.shape)
df.head()
```

```
(8, 10)
CPU times: user 29 ms, sys: 2.75 ms, total: 31.8 ms
Wall time: 52.7 s
```

[20]:

	r1_area	r1_osm_id	r1_fclass	r1_name	r1_geometry	r2_area	r2_osm_id	r2_fclass	r2_name	r2_geometry
0	SYDNEY	2948478	footway	None	LINESTRING (151.21487 -33.87012, 151.21488 -33…	SYDNEY	143528258			
1	ALEXANDRIA	194616057	service	None	LINESTRING (151.18819 -33.91986, 151.18909 -33…	ALEXANDRIA	194616059			
2	ALEXANDRIA	194616059	service	None	LINESTRING (151.18904 -33.92017, 151.18816 -33…	ALEXANDRIA	194616057			
3	ALEXANDRIA	194616059	service	None	LINESTRING (151.18904 -33.92017, 151.18816 -33…	MASCOT	194616057			
4	DARLINGTON	209795720	service	None	LINESTRING (151.19407 -33.89044, 151.19422 -33…	DARLINGTON	209795721			

```
0   footway    None  LINESTRING(151.2127419 -33.8736134,151.212755 …
1   service    None  LINESTRING(151.1890405 -33.9201711,151.1881568…
2   service    None  LINESTRING(151.1881929 -33.9198573,151.189086 …
3   service    None  LINESTRING(151.1881929 -33.9198573,151.189086 …
4   service    None  LINESTRING(151.194175 -33.8904839,151.1941246 …
```

[21]:
```
df['r1_osm_id'].isin(df['r2_osm_id']).value_counts()
```

[21]:
```
True    8
Name: r1_osm_id, dtype: int64
```

[22]:
```
df1 = df[[col for col in df.columns if col.startswith('r1')]].copy()
df1.rename(columns={'r1_geometry': 'geometry'}, inplace=True)
df2 = df[[col for col in df.columns if col.startswith('r2')]].copy()
df2['r2_geometry'] = df2['r2_geometry'].apply(wkt.loads)
df2.rename(columns={'r2_geometry': 'geometry'}, inplace=True)
df2 = gpd.GeoDataFrame(df2, geometry='geometry')
```

[23]:
```
map2 = KeplerGl()
map2.add_data(df1, name='roads1')
map2.add_data(df2, name='roads2')
map2.save_to_html(file_name='map2.html')
```

```
User Guide: https://docs.kepler.gl/docs/keplergl-jupyter
Map saved to map2.html!
```

[ ]: