Washington

October 13, 2021

[2]:
```
%load_ext autoreload
%autoreload 2
import sys; sys.path.append('../')

# Prettymaps
from prettymaps import *
# Vsketch
import vsketch
# OSMNX
import osmnx as ox
# Matplotlib-related
import matplotlib.font_manager as fm
from matplotlib import pyplot as plt
from descartes import PolygonPatch
# Shapely
from shapely.geometry import *
from shapely.affinity import *
from shapely.ops import unary_union
```

[3]:
```
# Style parameters
palette = ['#FFC857', '#E9724C', '#C5283D']
background_c = '#F2F4CB'
dilate = 300

# Setup figure
fig, ax = plt.subplots(figsize = (10, 10), constrained_layout = True)

# Plot
layers = plot(
    (38.89872,-77.03654),
    radius = 1300,
    ax = ax,
    layers = {
        'perimeter': {'circle': False, 'dilate': dilate},
        'streets': {
            'width': {
                    'motorway': 5,
                'trunk': 5,
                'primary': 4.5,
                'secondary': 4,
                'tertiary': 3.5,
                'residential': 3,
                'service': 2,
                'unclassified': 2,
                'pedestrian': 2,
                'footway': 1,
        },
        'circle': False,
        'dilate': dilate
    },
    'building': {
        'tags': {
            'building': True,
            'landuse': 'construction'
        },
        'union': False,
        'circle': False,
        'dilate': dilate
    },
    'water': {
        'tags': {
            'natural': ['water', 'bay']
        },
        'union': False,
        'circle': False,
        'dilate': dilate
    },
    'green': {
        'tags': {
            'landuse': ['grass', 'natrual'],
            'natural': ['island', 'wood'],
            'leisure': 'park'
        },
        'circle': False,
        'dilate': dilate
    },
    'forest': {
        'tags': {
            'landuse': 'forest'
        },
        'circle': False,
        'dilate': dilate
    },
    'parking': {
            'tags': {
                'amenity': 'parking', 'highway': 'pedestrian', 'man_made':␣
↪'pier'
            },
            'circle': False,
            'dilate': dilate
        },
    },

            drawing_kwargs = {
            'background': {'fc': '#F2F4CB', 'ec': '#dadbc1', 'hatch': 'ooo...',␣
↪'zorder': -1},
            'perimeter': {'fill': False, 'lw': 0, 'zorder': 0},
            'green': {'fc': '#D0F1BF', 'ec': '#2F3737', 'lw': 1, 'zorder': 1},
            'forest': {'fc': '#64B96A', 'ec': '#2F3737', 'lw': 1, 'zorder': 1},
            'water': {'fc': '#a1e3ff', 'ec': '#2F3737', 'hatch': 'ooo...',␣
↪'hatch_c': '#85c9e6', 'lw': 1, 'zorder': 2},
            'parking': {'fc': '#F2F4CB', 'ec': '#2F3737', 'lw': 1, 'zorder': 3},
            'streets': {'fc': '#2F3737', 'ec': '#475657', 'alpha': 1, 'lw': 0,␣
↪'zorder': 3},
            'building': {'palette': palette, 'ec': '#2F3737', 'lw': .5,␣
↪'zorder': 4},

        },

    osm_credit = {'x': .02, 'y': .05, 'color': '#2F3737'}
)

# Set bounds
xmin, ymin, xmax, ymax = layers['perimeter'].bounds
dx, dy = xmax-xmin, ymax-ymin
ax.set_xlim(xmin-.06*dx, xmax+.06*dx)
ax.set_ylim(ymin-.06*dy, ymax+.06*dy)

# Draw left text
#ax.text(
 #   xmin-.06*dx, ymin+.5*dy,
  #  'Barcelona, Spain',
   # color = '#2F3737',
   # rotation = 90,
   # fontproperties = fm.FontProperties(fname = '../assets/Permanent_Marker/
↪PermanentMarker-Regular.ttf', size = 35),
#)
# Draw top text
ax.text(
    xmin+0*dx, ymin-.05*dy,
    "White House, Washington",
    color = '#2F3737',
    fontproperties = fm.FontProperties(fname = '../assets/Permanent_Marker/
↪PermanentMarker-Regular.ttf', size = 35),
)

plt.savefig('../prints/washington.png')
plt.savefig('../prints/washington.svg')
```

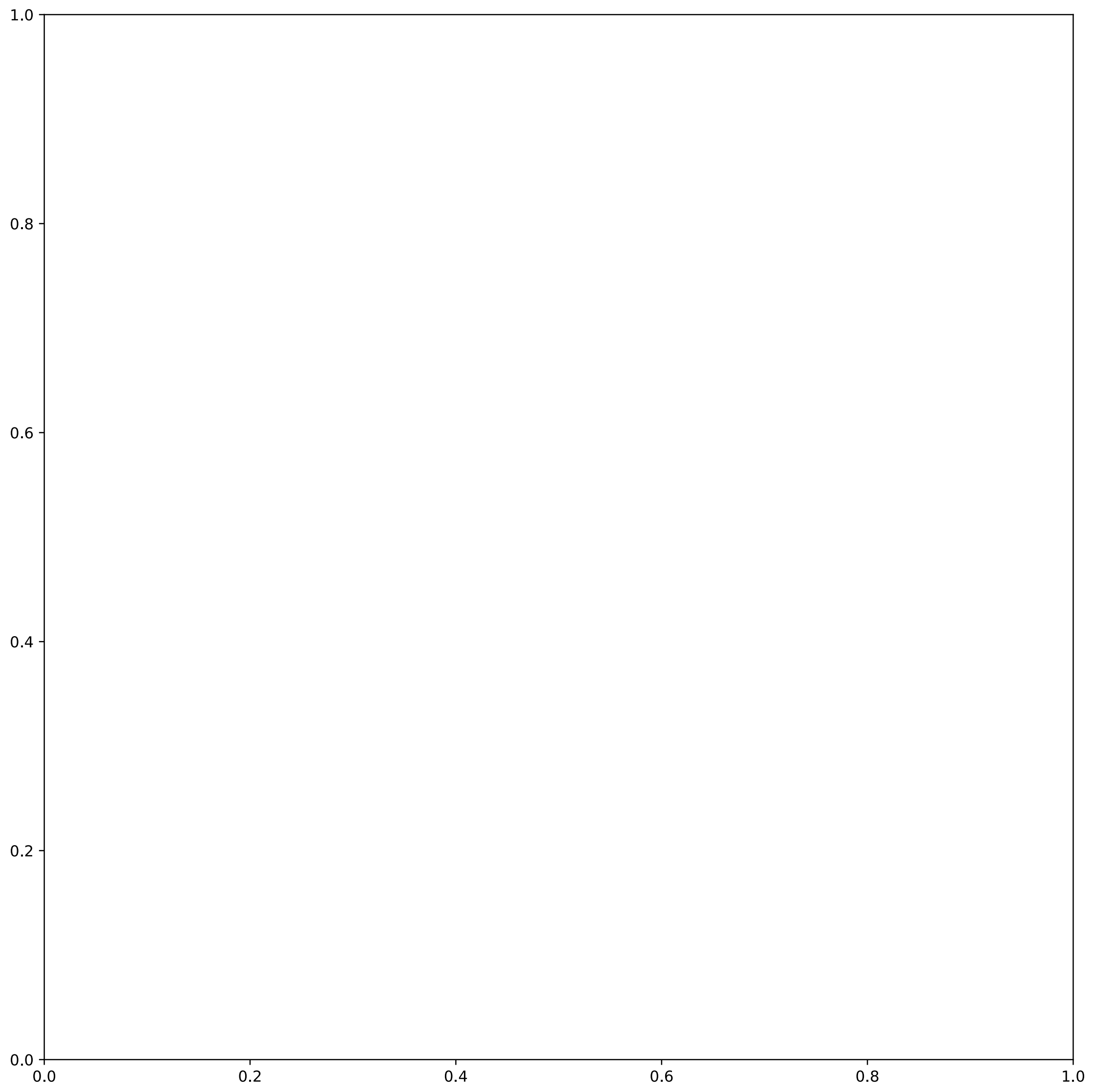
```
ERROR:shapely.geos:TopologyException: depth mismatch at  at 322730 4307000
```

```
Error No Shapely geometry can be created from null value
Offending object: footway 1
```

```
ERROR:shapely.geos:TopologyException: depth mismatch at  at 322730 4307000
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-3-e14085092522> in <module>
      8
      9 # Plot
---> 10 layers = plot(

     11     (38.89872,-77.03654),
     12     radius = 1300,

/opt/conda/lib/python3.9/site-packages/prettymaps/draw.py in plot(query, backup␣
↪postprocessing, radius, layers, drawing_kwargs, osm_credit, figsize, ax,␣
↪title, vsketch, x, y, scale_x, scale_y, rotation)
    234
    235         # Fetch layers
--> 236         layers = {

    237             layer: get_layer(
    238                 layer, **base_kwargs, **(kwargs if type(kwargs) == dict␣
↪else {})

/opt/conda/lib/python3.9/site-packages/prettymaps/draw.py in <dictcomp>(.0)
    235         # Fetch layers
    236         layers = {
--> 237             layer: get_layer(

    238                 layer, **base_kwargs, **(kwargs if type(kwargs) == dict␣
↪else {})
    239             )

/opt/conda/lib/python3.9/site-packages/prettymaps/fetch.py in get_layer(layer,␣
↪**kwargs)
    412     # Fetch streets or railway
    413     if layer in ["streets", "railway", "waterway"]:
```

```
--> 414         return get_streets(**kwargs, layer=layer)
    415     # Fetch Coastline
    416     elif layer == "coastline":

/opt/conda/lib/python3.9/site-packages/prettymaps/fetch.py in␣
 ↪get_streets(perimeter, point, radius, layer, width, custom_filter, buffer,␣
 ↪retain_all, circle, dilate, truncate_by_edge)
    338
    339         streets = unary_union(
--> 340             [
    341                 # Dilate streets of each highway type == 'highway' usin␣
 ↪width 'w'
    342                 MultiLineString(

/opt/conda/lib/python3.9/site-packages/prettymaps/fetch.py in <listcomp>(.0)
    340             [
    341                 # Dilate streets of each highway type == 'highway' usin␣
 ↪width 'w'
--> 342                 MultiLineString(
    343                     streets[
    344                         [highway in value for value in streets[layer]]

/opt/conda/lib/python3.9/site-packages/shapely/geometry/base.py in buffer(self,␣
 ↪distance, resolution, quadsegs, cap_style, join_style, mitre_limit,␣
 ↪single_sided)
    635             self._lgeos.GEOSBufferParams_setQuadrantSegments(params, re )
    636             self._lgeos.GEOSBufferParams_setSingleSided(params,␣
 ↪single_sided)
--> 637             return geom_factory(self.impl['buffer_with_params'](self,␣
 ↪params, distance))
    638
    639         if cap_style == CAP_STYLE.round and join_style == JOIN_STYLE.
 ↪round:

/opt/conda/lib/python3.9/site-packages/shapely/geometry/base.py in␣
 ↪geom_factory(g, parent)
     76     # Abstract geometry factory for use with topological methods below
     77     if not g:
---> 78         raise ValueError("No Shapely geometry can be created from null␣
 ↪value")
     79     ob = BaseGeometry()
     80     geom_type = geometry_type_name(g)

ValueError: No Shapely geometry can be created from null value
```


0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0

```
[ ]:
```