slides

May 22, 2019

[1]:
```
import os
import sys

print('Python: {}\n'.format(sys.version))
print('Currrent dir:', os.path.abspath(os.path.curdir))

def add_to_sys_path(this_path, up=False):
    """
    Prepend this_path to sys.path.
    If up=True, path refers to parent folder (1 level up).
    """
    for p in sys.path:
        p = os.path.abspath(p)
    if up:
        newp = os.path.abspath(os.path.join(this_path, '..'))
    else:
        newp = os.path.abspath(this_path)

    if this_path not in (p, p + os.sep):
        print('Path added to sys.path: {}'.format(newp))
        sys.path.insert(0, newp)

# if notebook inside another folder, eg ./notebooks:
up =  os.path.abspath(os.path.curdir).endswith('notebooks')
add_to_sys_path(os.path.curdir, up)

import numpy as np
import pandas as pd

from pprint import pprint as pp
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"

from IPython.display import HTML, Markdown, Image, IFrame
display(HTML("<style>.container { width:98% !important; }</style>"))

def new_section(title):
```

```
    style = "text-align:center;background:#c2d3ef;padding:20px;color:#ffffff;
 ↪font-size:3em;width:98%"
    return HTML('<div style="{}">{}</div>'.format(style, title))

%load_ext autoreload
%autoreload 2
```

```
Python: 3.6.7 (default, Feb 28 2019, 07:28:18) [MSC v.1900 64 bit (AMD64)]

Currrent dir: C:\Users\catch\Documents\GitHub\Geocoders_presentation\notebooks
Path added to sys.path: C:\Users\catch\Documents\GitHub\Geocoders_presentation

<IPython.core.display.HTML object>
```

[2]:
```
import geopy
import geopandas as gpd
import pycrs
import pyproj

import mpl_toolkits.basemap as mplmap
from mpl_toolkits.basemap import Basemap
import matplotlib.pyplot as plt

# project imports
from GeocodersComparison import (comparison as GeoComp4,
                                 inspect_geocoders,
                                 gc4utils,
                                 gc4settings)

# Directories:
dir_geo = GeoComp4.DIR_GEO
dir_html = GeoComp4.DIR_HTML
dir_img = GeoComp4.DIR_IMG
dir_rpt = GeoComp4.DIR_RPT

dir_raw = os.path.join(dir_geo, 'rawjson')

# keys from .env
GOOGLE_KEY = GeoComp4.GOOGLE_KEY
AZURE_KEY = GeoComp4.AZURE_KEY
W3W_dict = GeoComp4.W3W_dict
```

```
Places queried, var query_lst:
['New York City, NY, USA', "Cleopatra's needle, Central Park, New York, NY,
USA", 'Bronx county, NY, USA', 'Kings county, NY, USA', 'New York county, NY,
```

```
USA', 'Queens county, NY, USA', 'Richmond county, NY, USA', 'Boston, MA, USA']

Fetching API keys from environment file if found.
Module import operations: COMPLETE.
```

[3]:
```
geocs_dict = inspect_geocoders.get_dict_geocs_class()
n_geocs = len(geocs_dict)

geocs_reqs_dict = inspect_geocoders.get_dict_geocs_required_params(geocs_dict)
n_geocs_with_reqs = len(geocs_reqs_dict)

# Print basic info + test on new reqs:
#print(inspect_geocoders.inspect_geopy_geocs(geocs_dict, geocs_reqs_dict))

# def get_geocs_reqs_df(geocs_dict, geocs_reqs_dict, load_from_dict=True)
# the function checks for json file (geocs_class_reqs.json) holding a previous
# check & loads it if found;
# Pass load_from_dict=False to get a fresh test.

df_inspect, geocs_class_reqs = inspect_geocoders.get_geocs_reqs_df(geocs_dict,␣
 ↪geocs_reqs_dict)

inspect_rpt = inspect_geocoders.save_inspect_report(df_inspect, n_geocs,␣
 ↪n_geocs_with_reqs)
#print(inspect_rpt)
```

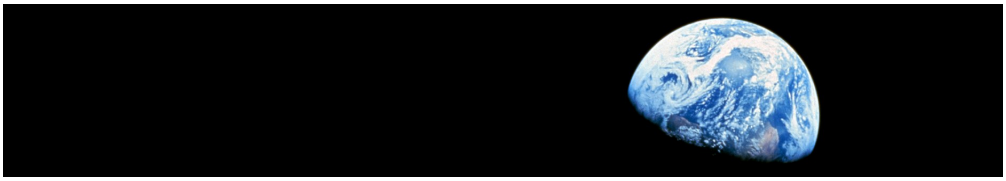
```
C:\Users\catch\Documents\GitHub\Geocoders_presentation\GeocodersComparison\repor
t\geocs_class_reqs.json
C:\Users\catch\Documents\GitHub\Geocoders_presentation\GeocodersComparison\repor
t\geocs_inspect.html
```

[4]:
```
# set defaults for all geocoders
geopy.geocoders.options.default_user_agent = 'this_app/1'
geopy.geocoders.options.default_timeout = 5

Man = "Manhattan, New York, NY USA"

goo = geopy.geocoders.GoogleV3(api_key=GOOGLE_KEY)
az = geopy.geocoders.AzureMaps(subscription_key=AZURE_KEY)
w3w = geopy.geocoders.What3Words(api_key=W3W_dict['W3W_API'])
arc = geopy.geocoders.ArcGIS()

# get raw results

ans = az.geocode(Man)
az_man_raw = ans.raw
```

```
ans = goo.geocode(Man)
goo_man_raw = ans.raw

ans = arc.geocode(Man)
arc_man_raw = ans.raw
```

[5]:
```
# Query list:
# Each query list string is passed to the geocoding function.
query_lst = GeoComp4.query_lst
print("\nList of query strings that will be passed to each geocoder:")
for i, q in enumerate(query_lst):
    print('{}. {!r}'.format(i, q))

# Geoloders in the comparison:
colors_dict = GeoComp4.colors_dict

geocs = GeoComp4.geocs
#print("\nGeocoders compared:\n", geocs)

# the relative path is needed for IFrame
fname_boston_rel = os.path.relpath(os.path.join(dir_html, 'Boston.html'))
fname_nyc_rel = os.path.relpath(os.path.join(dir_html, 'New_York_City.html'))
fname_kings_rel = os.path.relpath(os.path.join(dir_html, 'Kings_county.html'))

globe_pm_eq = os.path.join(dir_img, 'globe_pm_eq.png')
projs_mercator = os.path.join(dir_img, 'projs_mercator.png')
projs_cylindrical = os.path.join(dir_img, 'projs_cylindrical.png')
projs_others = os.path.join(dir_img, 'projs_others.png')

rpt_geocs_inspect = os.path.join(dir_rpt, "geocs_inspect.html")
```

```
List of query strings that will be passed to each geocoder:
0. 'New York City, NY, USA'
1. "Cleopatra's needle, Central Park, New York, NY, USA"
2. 'Bronx county, NY, USA'
3. 'Kings county, NY, USA'
4. 'New York county, NY, USA'
5. 'Queens county, NY, USA'
6. 'Richmond county, NY, USA'
7. 'Boston, MA, USA'
```



Earthrise

1 Earth : N Geocoders

0.1 ## Geocoding Explorations with Python

0.1.1 Cat Chenal, Ph.D.

Postgres Women NYC, Renee Phillips, organizer, May, 23 2019

1 This happened to me:

[6]:
```
IFrame(fname_boston_rel, 1350, 600)
```

[6]:
```
<IPython.lib.display.IFrame at 0x1344f4a5240>
```

[7]:
```
IFrame(fname_kings_rel, 1350, 600)
```

[7]:
```
<IPython.lib.display.IFrame at 0x1344f4a5470>
```

2 What's in a map?

[8]:
```
inspect_geocoders.show_map_components()
```

[8]:
```
<IPython.core.display.HTML object>
```

Mapping with Python

Library	Description
GDAL	Fundamental package for processing vector and raster data formats (many modules below depend on this)
Shapely	Manipulation and analysis of planar geometric objects
Geopandas	Pandas + Shapely
Fiona	alternative for geopandas (shapefile loader)
Geopy	Geocoding library
Pyproj	Performs cartographic transformations and geodetic computations
PyCRS	"For reading, writing, and converting between various common coordinate reference system (CRS) string and data source formats"

Library	Description
Pysal	Library of spatial analysis functions written in Python
Folium	Wrapper for Leaflet: beautiful and highly customizable maps
OSMnx	OpenStreetMap street networks
Networkx	Network analysis and routing
Scipy.spatial	Spatial algorithms and data structures
Rtree	Spatial indexing for Python for quick spatial lookups
Cartopy	"Cartopy makes use of the powerful PROJ.4, NumPy and Shapely libraries and includes a programmatic interface built on top of Matplotlib for the creation of publication quality maps."
Matplotlib toolkit Basemap	
GeoViews	Interactive Maps for the web
Geoplot	High-level geospatial data visualization library for Python
Dash	Dash is a Python framework for building analytical web applications
More here:	https://automating-gis-processes.github.io/2017/lessons/L1/Intro-Python-GIS.html

2.1 # Full-blown Geographic Information System (GIS) platforms:

2.2 ESRI ArcGIS

2.3 ## QGIS (open source)

2.4 # GeoPandas

Geopy geocoders included! `geopandas.tools.geocode`

Get PostGIS data: `geopandas.read_postgis`

"Returns a GeoDataFrame corresponding to the result of the query string, which must contain a geometry column in WKB representation."

What's geocoding?

---

3 *Forward*: "query string" (longitude, latitude)

4 *Reverse*: (longitude, latitude) "address/description"

`Obtaining the geolocation coordinates of a location specified by query string can be achieved u`

[9]:
```
HTML("<h1>Accessing geocoding services:</h1>")
inspect_geocoders.show_geoc_access()
```

[9]:
```
<IPython.core.display.HTML object>
```

[9]:
```
<IPython.core.display.HTML object>
```

[10]:
```
HTML(filename=inspect_rpt)
```

[10]:
```
<IPython.core.display.HTML object>
```

4.0.1 PostGIS geocoder :`SELECT g.<this>, g.<that> FROM geocode(<query str>,1) AS g;`

4.0.2 Tiger Geocoder (extra PostGIS feature): Works with the TIGER (Topologically Integrated Geographic Encoding and Referencing system) and Line and Master Address database export released by the US Census Bureau.

4.0.3 Census Bureau geocoder: https://geocoding.geo.census.gov/

Caveat codor: If y = Forward(query), then Reverse(y) != query
(Only true for What3Words)

```
This company has implemented a solution to a rather common logistic problem: how to deliver goo

They tiled the land area with squares of 3 x 3 meters and assigned to each square a unique comb
```

What3Words
`W3W is better accessed with the usual `requests` library, if you need the grid (squares)`

4.0.4 The W3W API has an additional endpoint, the `grid`.

4.0.5 It is NOT available in `geopy.geocoders`.

inspect$_{g}eocoders.show_{w}3w_{e}ndpoints()$
```
W3W  map website has different map providers; the default: GoogleMaps

The result of a geocoding query is a 3x3 square named with 3 words

You can get the 3 words for each square in your appartment...
\end{verbatim}

    \begin{center}\rule{0.5\linewidth}{\linethickness}\end{center}

\hypertarget{their-map-website-is-using-googles-api-for-geocoding.}{%
\subsection{Their map website is using Google's API for
geocoding.}\label{their-map-website-is-using-googles-api-for-geocoding.}}

\hypertarget{this-means-that-they-obtain-the-geocoordinates-of-a-traditional-query-then-use-the
\subsubsection{\texorpdfstring{This means that they obtain the
geocoordinates of a traditional query, then use them to \emph{reverse}
code using the w3w
```

```
api.}{This means that they obtain the geocoordinates of a traditional query, then use them to r

\hypertarget{they-carry-over-the-quirky-results}{%
\subsubsection{They carry over the quirky
results\ldots{}}\label{they-carry-over-the-quirky-results}}

    ``Manhattan, New York, NY USA'' :: ``soil.pushes.mole'' ~~~ 225 CPW,
NY, NY 10024, a building on CPW, btw 82nd \& 83rd St

``New York county, NY, USA'' :: "dress.sharp.brave ~~~ City Hall

``Brooklyn, NY, USA'' :: ``recent.pints.giving'' ~~~ Intersection of
Atlantic \& Brooklyn aves

``Kings county, NY, USA'' :: ``recent.pints.giving'' ~~~ same

    \hypertarget{calm.spot.green}{%
\subsection{``calm.spot.green''}\label{calm.spot.green}}

\hypertarget{is-a-square-in-minnesota}{%
\subsubsection{\ldots is a square in
Minnesota}\label{is-a-square-in-minnesota}}

\hypertarget{web-site}{%
\subsection{Web site:}\label{web-site}}

\hypertarget{w3w-example-uses.noses.cried}{%
\subsubsection{W3W example:
``uses.noses.cried''}\label{w3w-example-uses.noses.cried}}
inspect_geocoders.show_w3w_nyc_results()
    { Why so many geocoding APIs?}

    They restrict their search areas to a specific country,
e.g.~\texttt{BANFrance}

They return their results in a specific language, e.g.~\texttt{Yandex}:
Russian, \texttt{Baidu}: `Chinese'

They use a different coordinate system: \texttt{What3Words}

They package their results in a different way

    \begin{tcolorbox}[breakable, size=fbox, boxrule=1pt, pad at break*=1mm,colback=cellbackgrou
\prompt{In}{incolor}{11}{\hspace{4pt}}
\begin{Verbatim}[commandchars=\\\{\}]
\PY{n}{HTML}\PY{p}{(}\PY{l+s+s2}{\PYZdq{}\PYZdq{}\PYZdq{}}\PY{l+s+s2}{\PYZlt{}h3\PYZgt{}This di
\PY{l+s+s2}{the `raw` results of a geocoding query\PYZlt{}/h3\PYZgt{}}
\PY{l+s+s2}{\PYZlt{}h4 style=}\PY{l+s+s2}{\PYZdq{}}\PY{l+s+s2}{text\PYZhy{}align:left;}\PY{l+s-
```

```
\PY{l+s+s2}{Raw results for query=}\PY{l+s+s2}{\PYZdq{}}\PY{l+s+si}{\PYZob{}\PYZcb{}}\PY{l+s+s

\PY{n}{HTML}\PY{p}{(}\PY{l+s+s1}{\PYZsq{}}\PY{l+s+s1}{\PYZlt{}h5 style=}\PY{l+s+s1}{\PYZdq{}}\
\PY{n}{pp}\PY{p}{(}\PY{n}{arc\PYZus{}man\PYZus{}raw}\PY{p}{)}
\PY{n}{HTML}\PY{p}{(}\PY{l+s+s1}{\PYZsq{}}\PY{l+s+s1}{\PYZlt{}h5 style=}\PY{l+s+s1}{\PYZdq{}}\
\PY{n}{pp}\PY{p}{(}\PY{n}{goo\PYZus{}man\PYZus{}raw}\PY{p}{)}
\end{Verbatim}
\end{tcolorbox}

            \begin{tcolorbox}[breakable, boxrule=.5pt, size=fbox, pad at break*=1mm, opacityfil
\prompt{Out}{outcolor}{11}{\hspace{3.5pt}}
\begin{Verbatim}[commandchars=\\\{\}]
<IPython.core.display.HTML object>
\end{Verbatim}
\end{tcolorbox}

            \begin{tcolorbox}[breakable, boxrule=.5pt, size=fbox, pad at break*=1mm, opacityfil
\prompt{Out}{outcolor}{11}{\hspace{3.5pt}}
\begin{Verbatim}[commandchars=\\\{\}]
<IPython.core.display.HTML object>
\end{Verbatim}
\end{tcolorbox}

    \begin{Verbatim}[commandchars=\\\{\}]
\{'address': 'Manhattan, New York',
 'attributes': \{\},
 'extent': \{'xmax': -73.99600999999994,
            'xmin': -74.01600999999995,
            'ymax': 40.72450000000004,
            'ymin': 40.704500000000046\},
 'location': \{'x': -74.00600999999995, 'y': 40.714500000000044\},
 'score': 100\}
\end{Verbatim}

            \begin{tcolorbox}[breakable, boxrule=.5pt, size=fbox, pad at break*=1mm, opacityfil
\prompt{Out}{outcolor}{11}{\hspace{3.5pt}}
\begin{Verbatim}[commandchars=\\\{\}]
<IPython.core.display.HTML object>
\end{Verbatim}
\end{tcolorbox}

    \begin{Verbatim}[commandchars=\\\{\}]
\{'address\_components': [\{'long\_name': 'Manhattan',
                          'short\_name': 'Manhattan',
                          'types': ['political',
                                    'sublocality',
                                    'sublocality\_level\_1']\},
                         \{'long\_name': 'New York',
```

```
                         'short\_name': 'New York',
                         'types': ['locality', 'political']\},
                        \{'long\_name': 'New York County',
                         'short\_name': 'New York County',
                         'types': ['administrative\_area\_level\_2', 'political']\},
                        \{'long\_name': 'New York',
                         'short\_name': 'NY',
                         'types': ['administrative\_area\_level\_1', 'political']\},
                        \{'long\_name': 'United States',
                         'short\_name': 'US',
                         'types': ['country', 'political']\}],
 'formatted\_address': 'Manhattan, New York, NY, USA',
 'geometry': \{'bounds': \{'northeast': \{'lat': 40.882214, 'lng': -73.907\},
                         'southwest': \{'lat': 40.6803955, 'lng': -74.047285\}\},
             'location': \{'lat': 40.7830603, 'lng': -73.9712488\},
             'location\_type': 'APPROXIMATE',
             'viewport': \{'northeast': \{'lat': 40.820045,
                                          'lng': -73.90331300000001\},
                           'southwest': \{'lat': 40.698078,
                                          'lng': -74.03514899999999\}\}\},
 'place\_id': 'ChIJYeZuBI9YwokRjMDs\_IEyCwo',
 'types': ['political', 'sublocality', 'sublocality\_level\_1']\}
\end{Verbatim}

    \hypertarget{a-locations-bounding-box-is-always-part-of-the-raw-output-i.e.-as-extents-bou
\subsection{A location's bounding box is always part of the raw output:
i.e.~as `extents', `bounds', or
`boundingBox'}\label{a-locations-bounding-box-is-always-part-of-the-raw-output-i.e.-as-extents-

\hypertarget{are-the-boxes-stored-or-calculated}{%
\subsection{Are the boxes stored or
calculated?}\label{are-the-boxes-stored-or-calculated}}

\begin{figure}
\centering
\includegraphics{../GeocodersComparison/images/comp_Loc_Center_tbl.svg}
\caption{Output of GeoComp4.df\_to\_pic(df\_T\_2,
save\_tbl\_name=`comp\_Loc\_Center\_tbl')}
\end{figure}

    \hypertarget{in-the-case-of-arcgis-it-seems-the-boxes-are-stored-r-tree-boxes}{%
\subsection{In the case of ArcGis, it seems the boxes are stored; R-Tree
boxes?}\label{in-the-case-of-arcgis-it-seems-the-boxes-are-stored-r-tree-boxes}}

\hypertarget{for-nominatim-and-google-a-location-coordinates-are-different-from-the-box-center}
\subsection{For Nominatim and Google, a location coordinates are
\textasciitilde{} different from the box
center}\label{for-nominatim-and-google-a-location-coordinates-are-different-from-the-box-cente
```

```
    \hypertarget{knowing-all-this-is-not-enough-to-find-out-the-quirks-in-my-results-need-to-g
\section{Knowing all this is not enough to find out the quirks in my
results: need to get to the GIS of
it!}\label{knowing-all-this-is-not-enough-to-find-out-the-quirks-in-my-results-need-to-get-to-

    \begin{verbatim}
This is because maping is a science that tries to solve a transformation problem: 3D &rarr; 2D
\end{verbatim}

    \begin{tcolorbox}[breakable, size=fbox, boxrule=1pt, pad at break*=1mm,colback=cellbackgrou
\prompt{In}{incolor}{93}{\hspace{4pt}}
\begin{Verbatim}[commandchars=\\\{\}]
\PY{n}{HTML}\PY{p}{(}\PY{l+s+s1}{\PYZsq{}}\PY{l+s+s1}{\PYZlt{}h1\PYZgt{}3D \PYZam{}rarr; 2D\PY
\PY{n}{Image}\PY{p}{(}\PY{n}{globe\PYZus{}pm\PYZus{}eq}\PY{p}{)}
\end{Verbatim}
\end{tcolorbox}

            \begin{tcolorbox}[breakable, boxrule=.5pt, size=fbox, pad at break*=1mm, opacityfil
\prompt{Out}{outcolor}{93}{\hspace{3.5pt}}
\begin{Verbatim}[commandchars=\\\{\}]
<IPython.core.display.HTML object>
\end{Verbatim}
\end{tcolorbox}


\prompt{Out}{outcolor}{93}{}

    \begin{center}
    \adjustimage{max size={0.9\linewidth}{0.9\paperheight}}{output_39_1.png}
    \end{center}
    { \hspace*{\fill} \\}


    \begin{tcolorbox}[breakable, size=fbox, boxrule=1pt, pad at break*=1mm,colback=cellbackgrou
\prompt{In}{incolor}{136}{\hspace{4pt}}
\begin{Verbatim}[commandchars=\\\{\}]
\PY{n}{HTML}\PY{p}{(}\PY{l+s+s1}{\PYZsq{}}\PY{l+s+s1}{\PYZlt{}H1\PYZgt{}Sometimes we want this
\PY{n}{Image}\PY{p}{(}\PY{n}{projs\PYZus{}cylindrical}\PY{p}{)}
\end{Verbatim}
\end{tcolorbox}

            \begin{tcolorbox}[breakable, boxrule=.5pt, size=fbox, pad at break*=1mm, opacityfil
\prompt{Out}{outcolor}{136}{\hspace{3.5pt}}
\begin{Verbatim}[commandchars=\\\{\}]
<IPython.core.display.HTML object>
\end{Verbatim}
\end{tcolorbox}
```

```
\prompt{Out}{outcolor}{136}{}

    \begin{center}
    \adjustimage{max size={0.9\linewidth}{0.9\paperheight}}{output_40_1.png}
    \end{center}
    { \hspace*{\fill} \\}


    \begin{tcolorbox}[breakable, size=fbox, boxrule=1pt, pad at break*=1mm,colback=cellbackgrou
\prompt{In}{incolor}{138}{\hspace{4pt}}
\begin{Verbatim}[commandchars=\\\{\}]
\PY{n}{HTML}\PY{p}{(}\PY{l+s+s1}{\PYZsq{}}\PY{l+s+s1}{\PYZlt{}H1\PYZgt{}Sometimes we want that
\PY{n}{Image}\PY{p}{(}\PY{n}{projs\PYZus{}mercator}\PY{p}{)}
\end{Verbatim}
\end{tcolorbox}

            \begin{tcolorbox}[breakable, boxrule=.5pt, size=fbox, pad at break*=1mm, opacityfil
\prompt{Out}{outcolor}{138}{\hspace{3.5pt}}
\begin{Verbatim}[commandchars=\\\{\}]
<IPython.core.display.HTML object>
\end{Verbatim}
\end{tcolorbox}


\prompt{Out}{outcolor}{138}{}

    \begin{center}
    \adjustimage{max size={0.9\linewidth}{0.9\paperheight}}{output_41_1.png}
    \end{center}
    { \hspace*{\fill} \\}


    \begin{tcolorbox}[breakable, size=fbox, boxrule=1pt, pad at break*=1mm,colback=cellbackgrou
\prompt{In}{incolor}{94}{\hspace{4pt}}
\begin{Verbatim}[commandchars=\\\{\}]
\PY{n}{HTML}\PY{p}{(}\PY{l+s+s1}{\PYZsq{}}\PY{l+s+s1}{\PYZlt{}H1\PYZgt{}Then again, sometimes w
\PY{n}{Image}\PY{p}{(}\PY{n}{projs\PYZus{}others}\PY{p}{)}
\end{Verbatim}
\end{tcolorbox}

            \begin{tcolorbox}[breakable, boxrule=.5pt, size=fbox, pad at break*=1mm, opacityfil
\prompt{Out}{outcolor}{94}{\hspace{3.5pt}}
\begin{Verbatim}[commandchars=\\\{\}]
<IPython.core.display.HTML object>
\end{Verbatim}
\end{tcolorbox}
```

```
\prompt{Out}{outcolor}{94}{}

    \begin{center}
    \adjustimage{max size={0.9\linewidth}{0.9\paperheight}}{output_42_1.png}
    \end{center}
    { \hspace*{\fill} \\}


    \hypertarget{there-are-28-other-possibilities}{%
\section{\ldots{} There are 28 other
possibilities\ldots{}}\label{there-are-28-other-possibilities}}

    {GIS Components}

\hypertarget{section}{%
\subsection{}\label{section}}
GIS linguo
Global Coordinate System: ellipsoid model + datum + geoid (gravity field model: global mean sea


Ellipsoid Models

Since the Earth is not a perfect sphere, we must account for the different length of its axes:
Speroids: major minor axes (m) reference
Clarke_1866  6378206.4  6356583.8  Mead's ranch in Kansas
GRS_1980     6378137    6356752.31414   Earth center
WGS_1984     6378137    6356752.31424518 ::  global ellipsoid model, geocentric, standard for G

WGS = World Geodetic System
GRS = Geodetic Reference System

Datum: surface benchmarks from surveys => network of standardized horizontal positions.
Any given spheroid has a smooth surface: not realistic (the earth's gravity field is not fixed)
"A datum is built on top of the selected spheroid, and can incorporate local variations in elev

NAD83 North America, ellipsoid: GRS80
ETRS89 Europe
OSGB36 Great Britain
JGD2011 Japanese Datum

List of Datum and spatial refs: https://spatialreference.org/ref/


[1] https://support.esri.com/en/technical-article/000006398
[2] https://geodesy.noaa.gov/datums/newdatums/index.shtml
.....................................................................Coordinate reference system
```

"The IOGPs EPSG Geodetic Parameter Dataset is a collection of definitions of coordinate refere

These codes show up when you examine the crs of a shapefile.
For example calling the crs method of the geopandas dataframes for NYC and Boston,we can see t
```
nyc_shp.crs {'init': 'epsg:2263'}
boston_shp.crs {'init': 'epsg:4326'}

EPSG:4326 belongs to the WGS84 GCS (lat, lon in DD)

```
IOGP: International Association of Oil & Gas Producers
EPSG: European Petroleum Survey Group

[1] http://www.epsg.org/
....# WKT and WKTB, Well-known

> Well-known Text (WKT) offers a compact machine- and human-readable representation of geometr
WKT may also be used for succinctly describing the critical elements of coordinate reference s
definitions.
=> POINT, LINESTRING, POLYGON and their MULTI version

EWKT and EWKB: Extended Well-Known Text/Binary the PostGIS-specific format:
This format includes the spatial reference system identifier (SRID) and up to 4 ordinate va
For example: SRID=4326;POINT(-44.3 60.1) to locate a longitude/latitude coordinate using the W
    Geodesics + Global Positioning System (GPS) =\textgreater{} relativity
(SR and GR)

The GPS ia an array of satellites orbiting the Earth which emit radio
signals along with the signal start time from their atomic clock. This
data and the speed of light is used by a ground receiver to obtain its
lat, lon and time with a (common) accuracy of \textasciitilde{} 5 meters
and local time to 40 billionths of a second{[}1{]}.

The correction that any GPS receiver applies to the data accounts for
two oposite, relativistic effects that do not cancel each other out.

This is because funny thinks happen when clocks are fast moving
(\textasciitilde14,000 km/hr) at an altitude where gravity is about 4
times weaker than on Earth (\textasciitilde20 km).

Special Relativity effect (time dilation viz c): T1 A fast moving clock
ticks more slowly (\textasciitilde6 microsec/day) General Relativity
effect (space and time curvature due to gravity): T2 The orbiting clock
ticks slightly faster, by about (\textasciitilde45 microsec/day).

```
Correction: T2 - T1 = \textasciitilde38 microsec/day (or about 10km/day)

{[}1{]} https://www.gps.gov/systems/gps/performance/accuracy/ {[}2{]}
http://www.themarginal.com/emc2/applications\_of\_relativity\_in\_gps.htm

    \hypertarget{precision-from-the-number-of-places-past-the-decimal-point}{%
\subsection{Precision from the number of places past the decimal
point}\label{precision-from-the-number-of-places-past-the-decimal-point}}
Decimal place (ordinal) Maximal resolved distance Comments on precision
1 11.1 km can distinguish the position of one large city from a neighboring large city
2 1.1 km can separate one village from the next
3 110 m can identify a large agricultural field or institutional campus
4 11 m can identify a parcel of land. It is comparable to the typical accuracy of an uncorrecte
5 1.1 m can distinguish trees from each other. Accuracy to this level with commercial GPS units
6 0.11 m can be used for laying out structures in detail, for designing landscapes, building ro
7 11 mm good for surveying and is near the limit of what GPS-based techniques can achieve
8 1.1 mm good for charting motions of tectonic plates and movements of volcanoes. Permanent, co
9 110 microns we are getting into the range of microscopy. For almost any conceivable applicati
> 9 useless indicates a computer or calculator was used and that no attention was paid to the i
    {Recap}

    \hypertarget{thanks}{%
\section{Thanks!}\label{thanks}}

\hypertarget{questions}{%
\section{Questions?}\label{questions}}

    \begin{tcolorbox}[breakable, size=fbox, boxrule=1pt, pad at break*=1mm,colback=cellbackgrou
\prompt{In}{incolor}{ }{\hspace{4pt}}
\begin{Verbatim}[commandchars=\\\{\}]

\end{Verbatim}
\end{tcolorbox}


    % Add a bibliography block to the postdoc
```

```
\relax
\providecommand\hyper@newdestlabel[2]{}
\providecommand\HyperFirstAtBeginDocument{\AtBeginDocument}
\HyperFirstAtBeginDocument{\ifx\hyper@anchor\@undefined
\global\let\oldcontentsline\contentsline
\gdef\contentsline#1#2#3#4{\oldcontentsline{#1}{#2}{#3}}
\global\let\oldnewlabel\newlabel
\gdef\newlabel#1#2{\newlabelxx{#1}#2}
\gdef\newlabelxx#1#2#3#4#5#6{\oldnewlabel{#1}{{#2}{#3}}}
\AtEndDocument{\ifx\hyper@anchor\@undefined
\let\contentsline\oldcontentsline
\let\newlabel\oldnewlabel
\fi}
\fi}
\global\let\hyper@last\relax
\gdef\HyperFirstAtBeginDocument#1{#1}
\providecommand*\HyPL@Entry[1]{}
\HyPL@Entry{0<</S/D>>}
\providecommand \oddpage@label [2]{}
\gdef \LT@i {\LT@entry
    {1}{226.78545pt}\LT@entry
    {1}{226.78545pt}}
\@writefile{lof}{\contentsline {figure}{\numberline {1}{\ignorespaces Earthrise\relax }}{5}{fig
\@writefile{toc}{\contentsline {subsection}{\numberline {0.1}\#\# Geocoding Explorations with P
\newlabel{geocoding-explorations-with-python}{{0.1}{5}{\#\# Geocoding Explorations with Python}
\@writefile{toc}{\contentsline {subsubsection}{\numberline {0.1.1}Cat Chenal, Ph.D.\\ \#\#\# Po
\newlabel{cat-chenal-ph.d.-postgres-women-nyc-renee-phillips-organizer-may-23-2019}{{0.1.1}{5}{
\@writefile{toc}{\contentsline {section}{\numberline {1}This happened to me:}{5}{section.1}\pro
\newlabel{this-happened-to-me}{{1}{5}{This happened to me:}{section.1}{}}
\@writefile{toc}{\contentsline {section}{\numberline {2}What's in a map?}{5}{section.2}\protect
\newlabel{whats-in-a-map}{{2}{5}{What's in a map?}{section.2}{}}
\@writefile{toc}{\contentsline {subsection}{\numberline {2.1}\# Full-blown Geographic Informati
\newlabel{full-blown-geographic-information-system-gis-platforms}{{2.1}{6}{\# Full-blown Geogra
\@writefile{toc}{\contentsline {subsection}{\numberline {2.2}ESRI ArcGIS}{6}{subsection.2.2}\pr
\newlabel{esri-arcgis}{{2.2}{6}{ESRI ArcGIS}{subsection.2.2}{}}
\@writefile{toc}{\contentsline {subsection}{\numberline {2.3}\#\# QGIS (open source)}{6}{subsec
\newlabel{qgis-open-source}{{2.3}{6}{\#\# QGIS (open source)}{subsection.2.3}{}}
\@writefile{toc}{\contentsline {subsection}{\numberline {2.4}\# GeoPandas}{6}{subsection.2.4}\p
\newlabel{geopandas}{{2.4}{6}{\# GeoPandas}{subsection.2.4}{}}
\@writefile{toc}{\contentsline {section}{\numberline {3}\emph  {Forward}: ``query string''  (lo
\newlabel{forward-query-string-longitude-latitude}{{3}{6}{\texorpdfstring {\emph {Forward}: ``q
\@writefile{toc}{\contentsline {section}{\numberline {4}\emph  {Reverse}: (longitude, latitude)
\newlabel{reverse-longitude-latitude-addressdescription}{{4}{6}{\texorpdfstring {\emph {Reverse
\@writefile{toc}{\contentsline {subsubsection}{\numberline {4.0.1}PostGIS geocoder :\texttt  {S
\newlabel{postgis-geocoder-select-g.this-g.that-from-geocodequery-str1-as-g}{{4.0.1}{7}{\texorp
\@writefile{toc}{\contentsline {subsubsection}{\numberline {4.0.2}Tiger Geocoder (extra PostGIS
\newlabel{tiger-geocoder-extra-postgis-feature-works-with-the-tiger-topologically-integrated-ge
\@writefile{toc}{\contentsline {subsubsection}{\numberline {4.0.3}Census Bureau geocoder: https
```