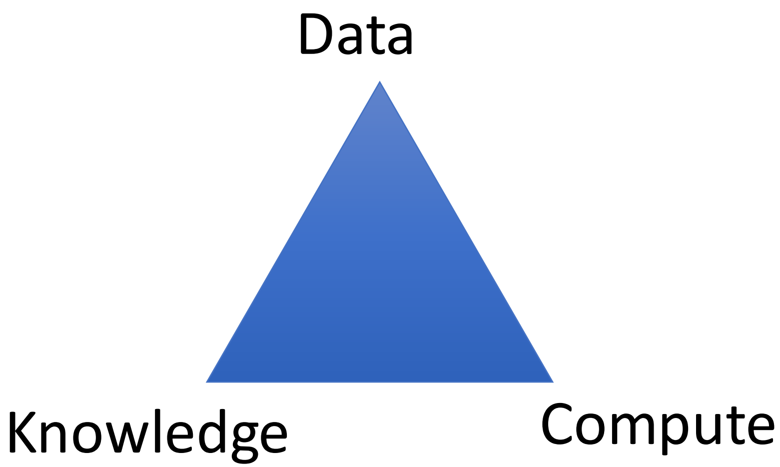
Geospatial Data Science
Content Block II: *Techniques*
Lecture 5
<u>Set Theory & Geospatial Queries</u>

Austin J. Brockmeier, Ph.D.

Monday, March 6th, 2023

Data Science



- Data holds answers
- Scientists have questions
- Computers give answers
- Data scientists know how to connect the three!

How many households in XYZ have access to healthy foods and outdoor spaces?

(In the near future)

User>> Please load the GIS data for XYZ with all household locations, all outdoor parks and recreation areas, grocery stores, and mobility routes among them.

Computer>> Interpreting instructions and executing commands…. relevant data loaded.

User>> Select the households from the XYZ data where the distance to the nearest grocery store is less than 1 mile and the distance to a public outdoor space is less than 0.25 miles.

Computer>> The query results in 123 households. How do you want me to visualize or organize the results?

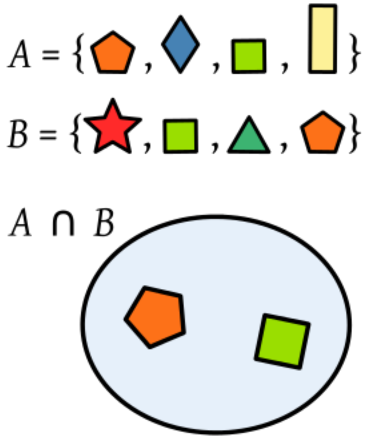
(Today)

User>> Hmmm. I’ve got to find some open data on households in XYZ. Maybe OpenStreetView has the parks and grocery stores…

I should have taken that 367 course to learn how to use Geopandas…

I wonder how GIS applications compute things under the hood.

<u>Outline</u>



Set Theory:

A mathematical language covering sets of points and relations among sets.

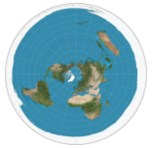
Concepts key to data science, probability, and statistics

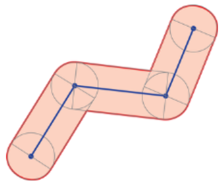
Combined with geometry allows a precise description of geospatial data

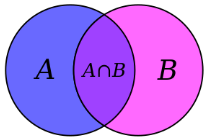
Relevant Scriptable/Programmable Computational Tools:

`Python`: programming language used broadly in data science practice and perfect for GDS

- Packages of relevant code
  - `Numpy` Numerical computing for Python
    - Mathematical functions on arrays (vectors, matrices, tensors) of numbers
  - `Geopandas` Combines GIS with data processing of `Pandas`
    - Coordinate awareness (can pre-project data)
    - https://geopandas.org/en/stable/community/ecosystem.html
  - `Shapely`
    - Compute on planar geometry (does not use geographic distances or elevations only 2D Euclidean space)
  - `Pandas` Python Data Analysis Library specifically for "<u>pan</u>el <u>da</u>ta"
    - spreadsheet-like <u>frame</u>work for computing and storing attributes of data in series







Outline

Computational environments:

Jupyter notebooks

Similar to interactive shell blocks for Mathematica, MATLAB , RStudio etc.

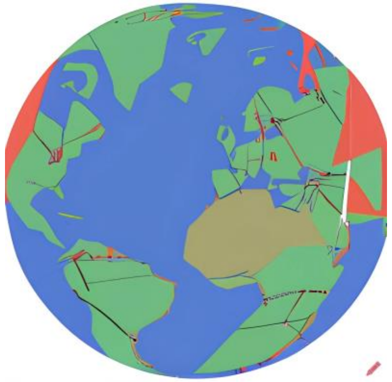
Google's Colaboratory runs most Jupyter notebooks from your web browser

Learning Tools:

Poll Everywhere

(Participation points!)

Geospatial data analysis assumptions (from Lectures 1–4)



Fact 1: The world is complex: need to simplify, approximate, and abstract for each task

Fact 2: Math provides precise definitions for abstract representations

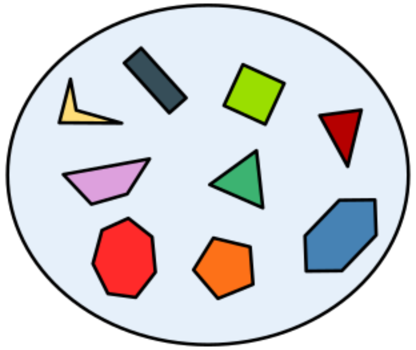
Fact 3: Digital data representations capture finite numerical precision about a finite set of points

- <u>Assumption 1:</u> Given an appropriate **scale,**
  points can adequately represent the location of people, objects, structures, cities, etc.;
  polygons or sets of polygons can represent areas;
  and spatial relationships can be locally approximated with planar geometry
- <u>Assumption 2:</u> Programming (written instructions to computers) is an efficient approach to answer questions from data.

- **<u>Conclusion</u>**: to learn geospatial data science techniques, we must understand the GIS principles, understand the underlying math, and be able to program/code

Set theory review

- A set is a collection of distinct things, called elements.
  - **Elements are assumed to be unique!**
  - Sets don’t keep track of how many times they appear.
- Sets don’t have a specific ordering
- An ordered list of the elements is a useful way of keeping track of set.



Python’s set and lists (and print)

```
a_list = [-3,-2,1,4,4,4,4]
a_set = set(a_list)
print(a_set)
```

```
[-3, -2, 1, 4]
```

Set theory review—cardinality: how to count!

Sets are ranked by cardinality—the number of elements

- Empty set: $\emptyset = \{\}$, and its cardinality (denoted $|\emptyset|$) is __________
- A singleton: $\mathcal{S} = \{\blacksquare\}$, and its cardinality (denoted $|\mathcal{S}|$) is __________
- Countable sets are iterable, where one can assign an index to each element. Countable sets can have finite or infinite cardinality
  - For the sets in Fig. 1 and Fig. 2, the cardinalities are _________ and _________

Fig. 1. A set of 3 elements.

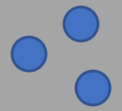
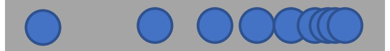


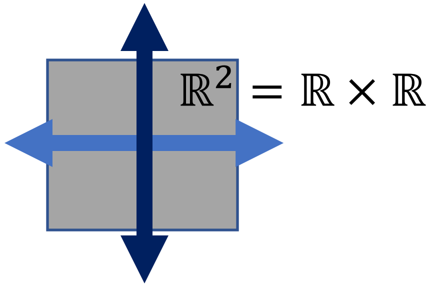
Fig. 2. An infinite geometric sequence.



- Subsets of Euclidean space are uncountable…
  - Finite spaces:
    - Line segment
    - Polygon
    - 3D volume

  - Infinite:
    - the real number line:
    - the 2D plane

$\mathbb{R}$



How to quantify the size of uncountable sets?

Set theory review—membership

The elements of a set are its members.

- E.g., if $\mathcal{S} = \{■\}$, then ■ is the only member and ■ $\in \mathcal{S}$. $\Delta \notin \mathcal{S}$.
- The empty set $\emptyset$ has no members.

Read '∈' as **is in.**

Read '∅' as **the empty set.**

Read '∉' as **is not in.**

Read '∧' as **and.**

Read ':' as **such that**

Building a set

1. Start with a base set $\mathcal{B}$
2. Specify some **logical** condition mathematically for filtering the membership for the new set

3. Notation

$$\mathcal{A} = \{x \in \mathcal{B} : x > 3\}$$

or $$\mathcal{A} = \{x : x \in \mathcal{B} \wedge x > 3\}$$

Python's list comprehension

```
B = set([-3,-2,1,4,4,4,4])

A = [x for x in B if x>3]
print(A)
```

```
[4]
```

Recall: Elements are assumed to be unique! Sets don't keep track of how many times they appear.

List comprehensions can involve function to make new lists

- Absolute value **function**: $\mathrm{abs}(x) = |x| = \begin{cases} x, & x \geq 0, \\ -x, & x < 0. \end{cases}$ *

```
print( [ abs(x) for x in [-3,0,2] ])
[3, 0, 2]
```

- Concatenate (into a tuple) to show the correspondence

```
print( [ (x, abs(x)) for x in [-3,0,2] ])
[(-3, 3), (0, 0), (2, 2)]
```

*Read as the absolute value of x is x if x is greater than or equal to 0, and negative x otherwise.

Set theory review—subsets

Read ‘⊆’ as **is a subset.**
Read ‘⊈’ as **is not a subset.**

Subset

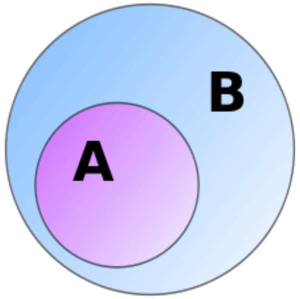
- A set $\mathcal{A}$ is a subset of another set $\mathcal{B}$ (denoted $\mathcal{A} \subseteq \mathcal{B}$) if every member in $\mathcal{A}$ is a member of $\mathcal{B}$
- E.g., if $\mathcal{A} = \{\Delta\},\ \mathcal{B} = \{\blacksquare, \Delta\}$ then $\mathcal{A} \subseteq \mathcal{B}$, but $\mathcal{B} \nsubseteq \mathcal{A}$
- Since $\emptyset$ has no members, then all of its (non-existence) members are members of any other set. This implies the $\emptyset \subseteq \mathcal{S}$ for any set $\mathcal{S}$.

Strict subset

- A set $\mathcal{A}$ is a strict subset of another set $\mathcal{B}$ (denoted $\mathcal{A} \subset \mathcal{B}$) if every member in $\mathcal{A}$ is a member of $\mathcal{B}$ and $\mathcal{A} \neq \mathcal{B}$

```python
# Basic set theory in Python

A = set(['v'])
B = set(['*', 'o', 'v'])


if all(x in B for x in A):
  print("A is subset of B")
else:
  print("A is not a subset of B")
```

```
A is subset of B
```



Set theory review—**logical** and **set** operations

Boolean algebra (important for queries!)

x,y are **logic**/Boolean variables

(either true or false)

Unary:

NOT x $\qquad \neg x$

Binary, symmetric:

x AND y $\qquad x \wedge y$

x OR y $\qquad x \vee y$

x XOR y = ((NOT x) AND y) OR (x AND (NOT y))

Binary, asymmetric:

x AND (NOT y)

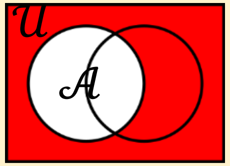
Set theory

$\mathcal{A}, \mathcal{B}$ are sets with $\mathcal{A}, \mathcal{B} \subseteq \mathcal{U}$

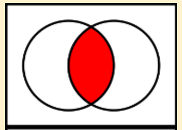
$\mathcal{U}$ is the universe of elements

Unary:

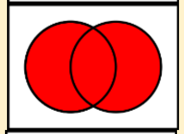
Complement: $\mathcal{A}^c = \{x \in \mathcal{U} : x \notin \mathcal{A}\}$



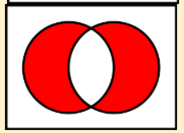
Binary, symmetric:



Intersect: $\mathcal{A} \cap \mathcal{B} = \{x \in \mathcal{U} : x \in \mathcal{A} \wedge x \in \mathcal{B}\}$

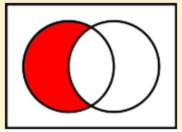


Union: $\mathcal{A} \cup \mathcal{B} = \{x \in \mathcal{U} : x \in \mathcal{A} \vee x \in \mathcal{B}\}$



Symmetric difference: $\mathcal{A} \,\Delta\, \mathcal{B}$

Binary, asymmetric:



Set difference: $\mathcal{A} \setminus \mathcal{B} = \{x \in \mathcal{A} : x \notin \mathcal{B}\}$

Set theory review

- If the intersection of two sets is the $\emptyset$, then the two sets are **disjoint**.

- In geospatial data, each member of a set (whether a point or higher dimensional objects) is associated with other attributes. The `GeoDataFrame` class from `Geopandas` can keep track of both.
  - The focus of Lab 6!

Poll Everywhere
(participation and gauge understanding)

Go to PollEv.com/ajbrock

(on your phone, computer, or other device)

Login with SSO (single sign-on) with your UD credentials

Q1. The set $\mathcal{S} = \{1,2,3,8,9,10\}$

What is the value of $|\mathcal{S}|$*, the cardinality of* $\mathcal{S}$*?*

A. 3
B. 6
C. ∞
D. $\{1,2,3,8,9,10\}$
E. 10

Q1. The set $\mathcal{S} = \{1,2,3,8,9,10\}$

What is the value of $|\mathcal{S}|$, the cardinality of $\mathcal{S}$?

A. 3
B. $\mathbf{6}$
C. ∞
D. $\{1,2,3,8,9,10\}$
E. 10

Q2. The set $\mathcal{S} = \{-1,0,3\}$

Which of these answers has the most number of true statements?

A. $3 \in \mathcal{S}$

B. $\emptyset \in \mathcal{S}$, $3 \in \mathcal{S}$

C. $\emptyset \notin \mathcal{S}$, $\{3\} \in \mathcal{S}$

D. $\emptyset \in \mathcal{S}$, $\{3\} \subseteq \mathcal{S}$, $\emptyset \subseteq \mathcal{S}$

E. $3 \in \mathcal{S}$, $\emptyset \subseteq \mathcal{S}$

Read '$\in$' as **in.**
Read '$\emptyset$' as **the empty set.**
Read '$\notin$' as **not in.**
Read '$\subseteq$' as **subset.**

Q2. The set $\mathcal{S} = \{-1,0,3\}$

Which of these answers has the most number of true statements?

A. $3 \in \mathcal{S}$

B. $\emptyset \in \mathcal{S}$, $\quad 3 \in \mathcal{S}$

C. $\emptyset \notin \mathcal{S}$, $\quad \{3\} \in \mathcal{S}$

D. $\emptyset \in \mathcal{S}$, $\quad \{3\} \subseteq \mathcal{S}$, $\quad \emptyset \subseteq \mathcal{S}$

E. $\mathbf{3} \in \boldsymbol{\mathcal{S}}$**,** $\quad \emptyset \subseteq \boldsymbol{\mathcal{S}}$

Read ‘∈’ as **in.**
Read ‘∅’ as **the empty set.**
Read ‘∉’ as **not in.**
Read ‘⊆’ as **subset.**

Q3. $\mathcal{S} = \{-1,0,3\}, \ \mathcal{A} = \{x \in \mathcal{S}: x = 0\}$

Which of these answers is true?

A. $\mathcal{A} = \{\emptyset\}$

B. $\mathcal{A} = \emptyset$

C. $\mathcal{A} = \{0\}$

D. $\mathcal{A} = 0$

Q3. $\mathcal{S} = \{-1,0,3\},\ \mathcal{A} = \{x \in \mathcal{S} : x = 0\}$

Which of these answers is true?

A. $\mathcal{A} = \{\emptyset\}$

B. $\mathcal{A} = \emptyset$

C. $\boldsymbol{\mathcal{A}} = \{\mathbf{0}\}$

D. $\mathcal{A} = 0$

Q3B. $\mathcal{S} = \{-1,0,3\},\ \mathcal{A} = \{x \in \mathcal{S}: x > 5\}$

Which of these answers is true?

A. $\mathcal{A} = \{\emptyset\}$
B. $\mathcal{A} = \emptyset$
C. $\mathcal{A} = \{0\}$
D. $\mathcal{A} = 0$

Q3B. $\mathcal{S} = \{-1,0,3\},\ \mathcal{A} = \{x \in \mathcal{S}: x > 5\}$

Which of these answers is true?

A. $\mathcal{A} = \{\emptyset\}$

B. $\boldsymbol{\mathcal{A}} = \emptyset$

C. $\mathcal{A} = \{0\}$

D. $\mathcal{A} = 0$

Q4. $\mathcal{S} = \{-1,0,3\},\ \mathcal{A} = \{[x, y]\colon\ x \in \mathcal{S}, y \in \mathcal{S}\}$

Which of these answers is true?

A. $\mathcal{A} = \{-1,0,3\}$

B. $\mathcal{A} = \{-1, -1, 0, 0, 3, 3\}$

C. $\mathcal{A} = \{[-1, -1], [-1,0], [-1,3], [0, -1], [0,0], [0,3], [3, -1], [3,0], [3,3]\}$

D. $\mathcal{A} = \{[-1, -1], [0,0], [3,3]\}$

Q4. $\mathcal{S} = \{-1,0,3\}$, $\mathcal{A} = \{[x, y]\colon\ x \in \mathcal{S}, y \in \mathcal{S}\}$

Which of these answers is true?

A. $\mathcal{A} = \{-1,0,3\}$

B. $\mathcal{A} = \{-1, -1, 0, 0, 3, 3\}$

C. $\boldsymbol{\mathcal{A}} = \{[\mathbf{-1}, \mathbf{-1}], [\mathbf{-1}, \mathbf{0}], [\mathbf{-1}, \mathbf{3}], [\mathbf{0}, \mathbf{-1}], [\mathbf{0}, \mathbf{0}], [\mathbf{0}, \mathbf{3}], [\mathbf{3}, \mathbf{-1}], [\mathbf{3}, \mathbf{0}], [\mathbf{3}, \mathbf{3}]\}$

D. $\mathcal{A} = \{[-1, -1], [0,0], [3,3]\}$

Q5. $f(x) = \begin{cases} \text{True}, & x \geq 0 \ \wedge x < 1, \\ \text{False}, & \text{otherwise.} \end{cases}$

Which of these statements (function evaluations) is true?

A. $f(2)$

B. $f(-2)$

C. $f(1) \vee f(0)$

D. $f(1) \wedge f(0)$

E. $\neg f(1)$

Q5. $f(x) = \begin{cases} \text{True}, & x \geq 0 \ \wedge x < 1, \\ \text{False}, & \text{otherwise.} \end{cases}$

Which of these statements (function evaluations) is true?

A. $f(2)$

B. $f(-2)$

C. $\boldsymbol{f(1)} \vee \boldsymbol{f(0)}$

D. $f(1) \wedge f(0)$

E. $\neg \boldsymbol{f(1)}$

Q5. $f(x) = \begin{cases} \text{True}, & x \geq 0 \ \wedge x < 1, \\ \text{False}, & \text{otherwise.} \end{cases}$

Which of these statements (function evaluations) is true?

A. $f(2)$

B. $f(-2)$

C. $f(1) \vee f(0)$

D. $f(1) \wedge f(0)$

E. $\neg f(1)$

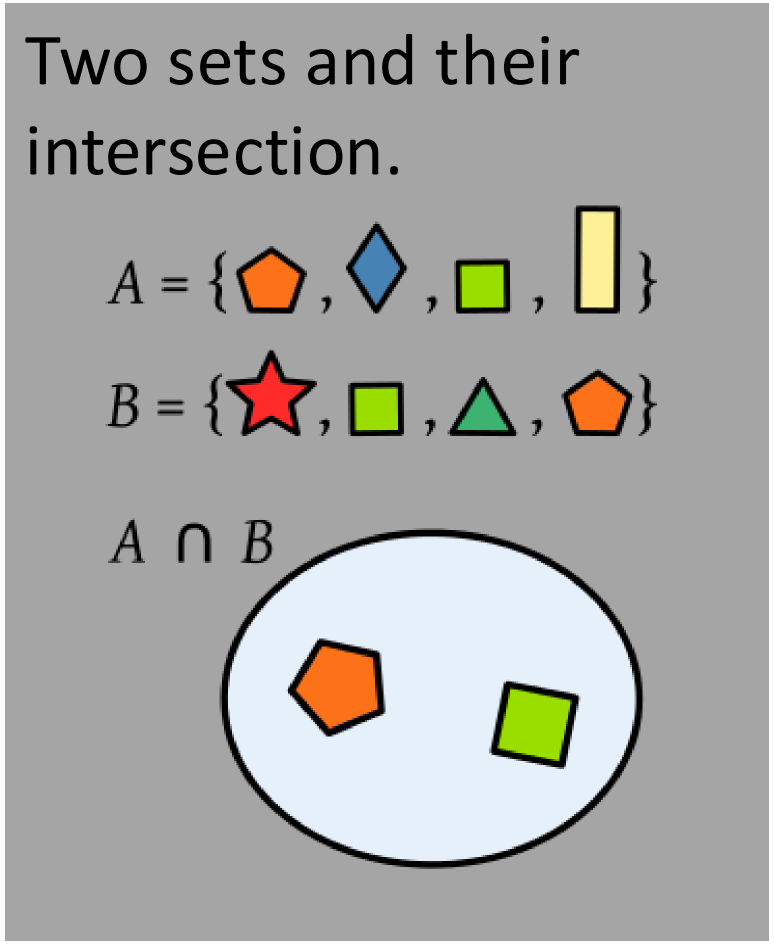
```
def f(x):
   return (x >= 0) and (x < 1)

logic_out_list = [f(2), f(-2), f(1) or f(0),
                  f(1) and f(0), not f(1) ]
answer_choices = 'ABCDE'

output = zip(answer_choices,logic_out_list)
print([ c+'.{}'.format(v) for c,v in output])
```

```
['A.False', 'B.False', 'C.True', 'D.False', 'E.True']
```

Q6. The figure depicts three sets for illustration.

Which of these statements is always true?

A. $(A \cap B) \cap A = A$
B. $(A \cap B) \cap A = B$
C. $(A \cap B) \cup A = A$
D. $(A \cap B) \cup A = B$
E. $(A \cap B) \cup A^c = B$



Q6. The figure depicts three sets for illustration.

Which of these statements is always true?

A. $(A \cap B) \cap A = A$

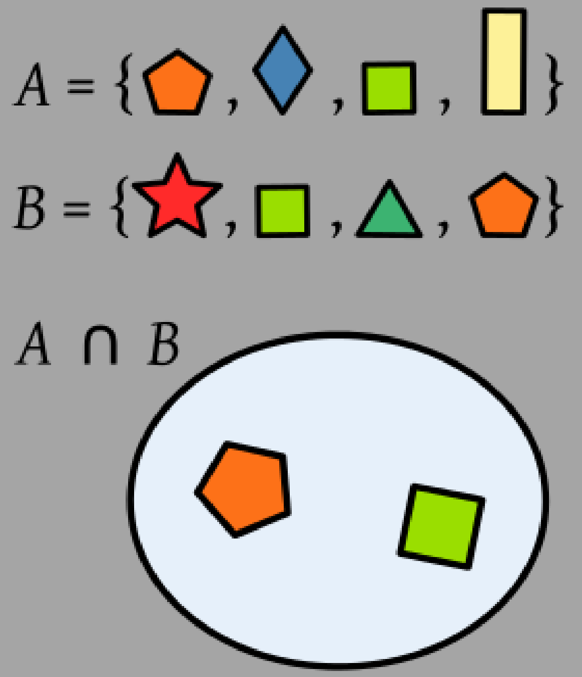
B. $(A \cap B) \cap A = B$

C. $\boldsymbol{(A \cap B) \cup A = A}$

D. $(A \cap B) \cup A = B$

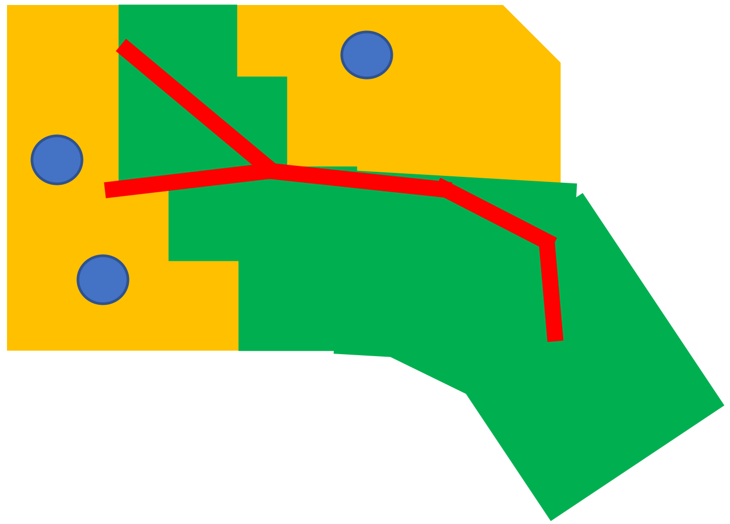
E. $(A \cap B) \cup A^c = B$

Two sets and their intersection.



Computing on geospatial sets in a planar/Euclidean space

- Additional properties (convexity) and operations
- Compute area
- Compute distances
  - between a point and a set
  - between two sets
  - extremes (shortest and longest) and typical within a set
- Simplify or approximate
  - remove fine details
  - cover an object with a coordinate-aligned bounding box
  - convex hull



- What is the total park area less than 1 mile from a particular household?

- Who lives on the boundaries of the area 12 miles from New Castle, Delaware? https://en.wikipedia.org/wiki/Twelve-Mile_Circle

- What are the set of points within a radius of 1 mile from any grocery store and less than 0.25 miles from any park?

- What are the set of points within a radius of 1 mile from any grocery store and farther than 0.25 miles from any park?

- What is the ‘metric diameter’ of the State of Delaware, i.e., what is the maximum distance between a pair of two points within the state over all possible pairs of points?

Mathematical notation for data and locations

$C \in \{-3, 0,2\}$ takes values from a discrete set

$a \in [0,1]$ is a number in the interval from 0 to 1.

$x \in \mathbb{R}$ is a real-valued **scalar** variable (temp., distance, elevation)

E.g., $x = 1.2235...$

$\boldsymbol{x} \in \mathbb{R}^2$, is a 2D **vector**

Equivalently, $\boldsymbol{x} = [x, y],\ x, y \in \mathbb{R}$

E.g., $\boldsymbol{x} = [-0.553, 0.1]$

$\begin{bmatrix} \boldsymbol{x}_0 \\ \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \boldsymbol{x}_3 \end{bmatrix} = [\boldsymbol{x}_i]_{i=0}^{3} \in \mathbb{R}^{4\times 2}$, is a **matrix**

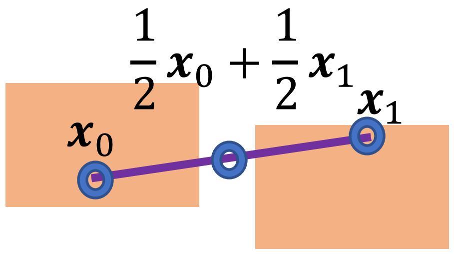
consisting of a series (a linear string) of four 2D vectors

```python
import numpy as np

x_coord,y_coord = -0.553, 0.1
x = np.array([x_coord,y_coord])
print(x)
```

```
[-0.553 0.1 ]
```

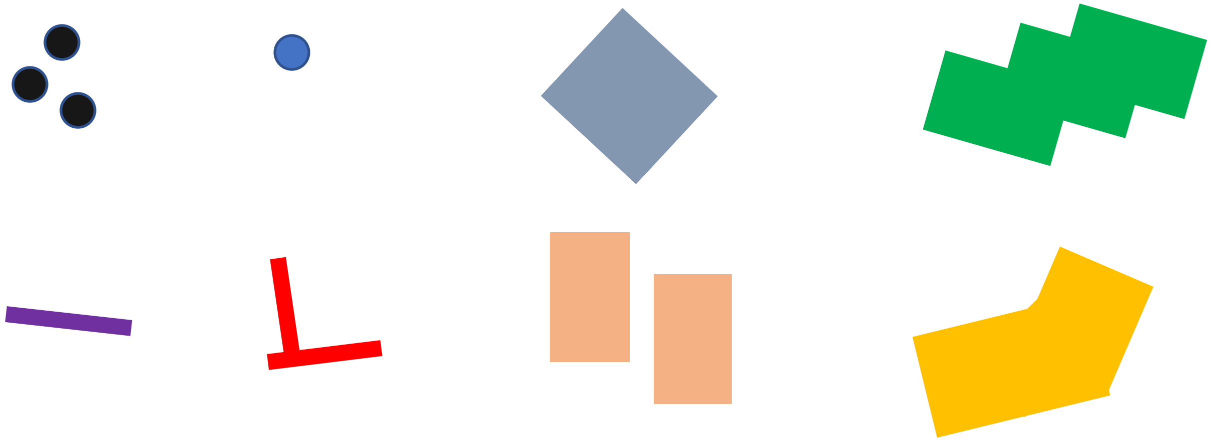
Convexity for sets of points in the plane

A set is convex if any point on a line segment between two points in the set is also in the set.



weighted average of the coordinates

- Which of these 8 sets are convex?



Distances from a point

Let d denote the Euclidean distance **function** between two points, defined as the 2-norm of the difference of the vectors

$$d(\boldsymbol{x}, \boldsymbol{x}') = \|\boldsymbol{x} - \boldsymbol{x}'\|_2 = \sqrt{(x - x')^2 + (y - y')^2}$$

where,

$\boldsymbol{x} = [x, y] \in \mathbb{R}^2$ and $\boldsymbol{x}' = [x', y'] \in \mathbb{R}^2$ are a 2D vectors

and $\|\boldsymbol{x}\|_2$ is the norm (distance from origin):

```
norm2 = np.sqrt(np.sum( x**2 ))
print(norm2)
```

Alternatively, the 1-Norm $\|\boldsymbol{x}\|_1 = |x| + |y|$ defines the

Manhattan distance $d_1(\boldsymbol{x}_1, \boldsymbol{x}_2) = |x - x'| + |y - y'|$

Buffers from a point

Combining set builder notation with geometry

Let $\boldsymbol{q} = [2, 1]$

$$\mathcal{S} = \{\boldsymbol{x} \in \mathbb{R}^2 \ : d(\boldsymbol{x}, \boldsymbol{q}) \leq 2\}$$

<u>What kind of geospatial set is $\mathcal{S}$?</u>

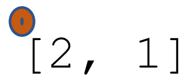
A) uncountable and infinite area

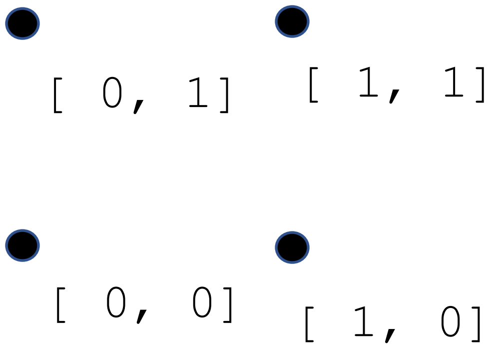
B) countable and infinite area

C) uncountable and finite area

D) countable and finite area

<u>Can you draw the geometry of $\mathcal{S}$?</u>





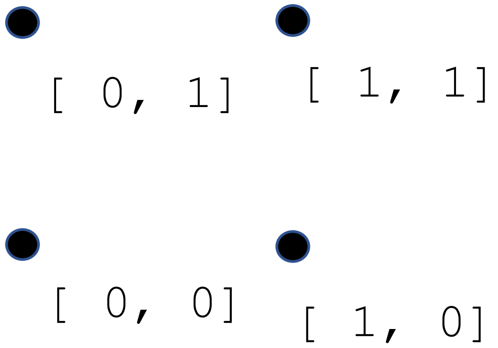
How to compute the distance to the origin for each?

$$\begin{bmatrix} \boldsymbol{x}_0 \\ \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \boldsymbol{x}_3 \end{bmatrix} = [\boldsymbol{x}_i]_{i=0}^{3} \in \mathbb{R}^{4\times 2},$$ is a **matrix**

consisting of a series (a linear string) of four 2D vectors

Note: A set of 4 points can define a path/curve, a ring, a polygon, or just 4 points!

Numpy axes/axis

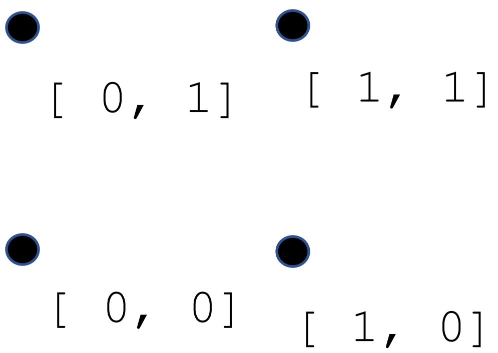



```
[[0 0]
 [0 1]
 [1 1]
 [1 0]]
```

```python
xs = np.array([ [ 0, 0], [0, 1], [1, 1], [1, 0] ])
origin = np.array([0,0])    # [0 0]
dists = np.sqrt(np.sum( (origin - xs)**2, axis=1))
# how we arrange data (rows or columns) affects axis :P
print(xs)

print(dists)
```

```
[0. 1. 1.41421356 1. ]
```

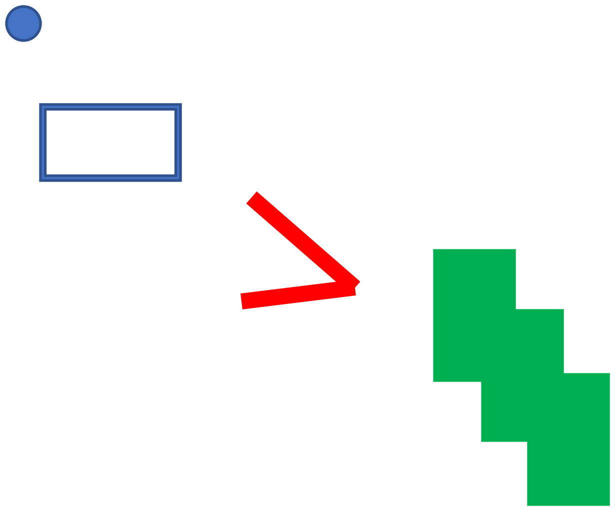
Numpy axes/axis and transpose T



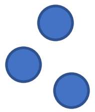

```
xs = np.array([ [ 0, 0], [0, 1], [1, 1], [1, 0] ]).T # matrix transpose
origin = np.array([0,0])
origin = origin[:, np.newaxis] # GOTCHA
dists2 = np.sqrt(np.sum( (origin - xs)**2, axis=0))
print(xs)
print(origin)
```

```
[[0 0 1 1] [0 1 1 0]]
[[0]
 [0]]
[0. 1. 1.41421356 1. ]
```

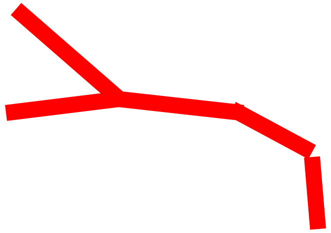
Geospatial sets in a planar/Euclidean space
`Shapely` classes

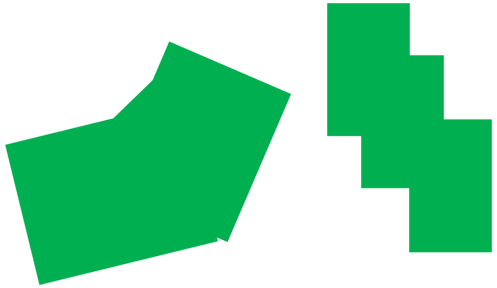


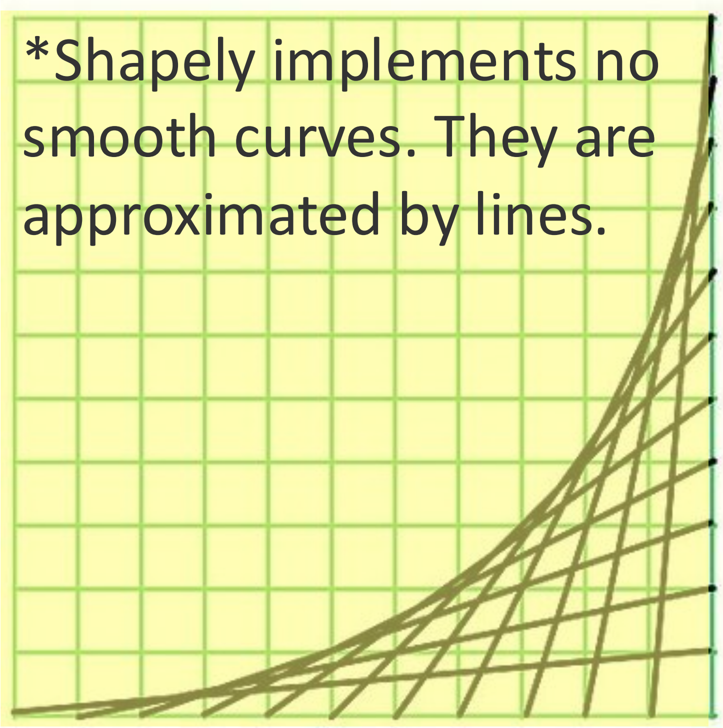
- Point `Point`
- Ring `LinearRing`
- Curve* `LineString`
- Surface `Polygon`



- Set of points `MultiPoint`
- Set of curves `MultiLineString`
- Set of surfaces `MultiPolygon`





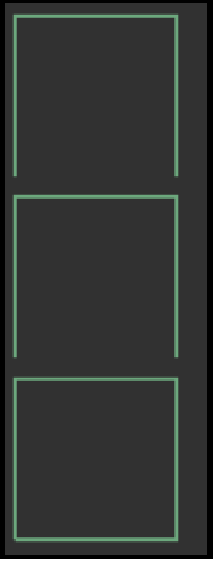


Defining geometric objects in Shapely

```
import numpy as np
from shapely import Point, LineString, LinearRing, Polygon

xs = np.array([ [ 0, 0], [0, 1], [1, 1], [1, 0] ])
points = [Point(x) for x in xs]
print(points)
```

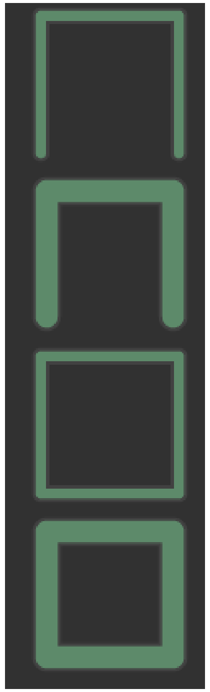
```
[<POINT (0 0)>, <POINT (0 1)>, <POINT (1 1)>, <POINT (1 0)>]
```

```
path = LineString(xs)
display(path)
path2 = LineString(points)
display(path2)
ring = LinearRing(points)
display(ring)
```



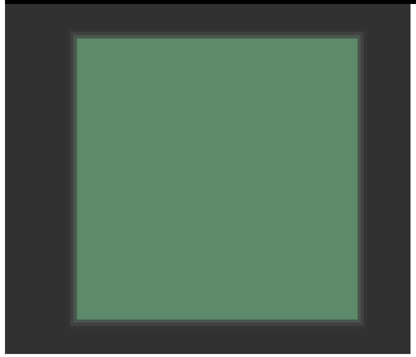
Defining geometric buffers in Shapely

```
display(path.buffer(0.05))
display(path.buffer(0.1))
display(ring.buffer(0.05))
display(ring.buffer(0.1))
```



Defining geometric objects in Shapely

```
square = Polygon(points)
display(square)
```



Computing areas w/ and w/o geometric buffers in Shapely

```
print(square.area)
print(square.buffer(0.1).area)
print(ring.area)
print(ring.buffer(0.1).area)
```

```
1.0
1.4313654849054596
0.0
0.7913654849054594
```

Accessing coordinates (and checking length)

```
point = Point(2,1)
print(list(point.coords))
print(point.coords[0][0])
```

```
[(2.0, 1.0)]
2.0
```

```
print(list(path.coords))
print(path.length)

print(list(ring.coords))
print(ring.length)
```

```
[(0.0, 0.0), (0.0, 1.0), (1.0, 1.0), (1.0, 0.0)]
3.0
[(0.0, 0.0), (0.0, 1.0), (1.0, 1.0), (1.0, 0.0), (0.0, 0.0)]
4.0
```

Manipulating coordinates (Polygon's with holes)

Polygon(*shell*[, *holes=None*])

```
small_ring = np.array(ring.xy).T /4
print(small_ring)



frame = Polygon(ring.coords,[small_ring+[0.1, 0.65],
                             small_ring+[0.65, 0.1] ])
```

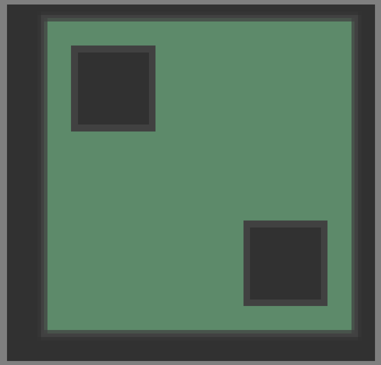
```
[[0.   0.  ]
 [0.   0.25]
 [0.25 0.25]
 [0.25 0.  ]
 [0.   0.  ]]
```

Manipulating coordinates (Polygon’s with holes)

Polygon(*shell*[, *holes=None*])

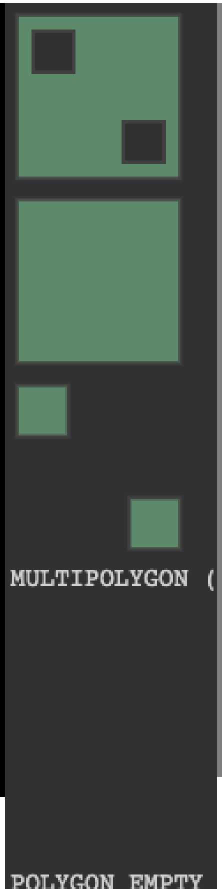
```
small_ring = np.array(ring.xy).T /4
print(small_ring)


frame = Polygon(ring.coords,[small_ring+[0.1, 0.65],
                             small_ring+[0.65, 0.1] ])
```

```
[[0.  0.  ]
 [0.  0.25]
 [0.25 0.25]
 [0.25 0.  ]
 [0.  0.  ]]
```



Set difference on polygons

```
display(frame)
display(square)

poly = square.difference(frame)
display(poly)
print(poly)

poly2 = frame.difference(square)
display(poly2)
print(poly2)
```



MULTIPOLYGON (((0.1 0.9, 0.35 0.9, 0.35 0.65, 0.1 0.65, 0.1 0.9)), ((0.65 0.35, 0.9 0.35, 0.9 0.1, 0.65 0.1, 0.65 0.35)))

POLYGON EMPTY

Upcoming Topics

- More interactions
- Plotting
- Regular grids

Action items

Try out the Slack!

Read more about set theory:

https://en.wikipedia.org/wiki/Set_(mathematics)

Familiarize with your computational environments:

Jupyter notebooks

Similar to interactive shell blocks for Mathematica, MATLAB , RStudio etc.

Google's Colaboratory runs Jupyter notebooks on the web

https://colab.research.google.com/

Refresh or learn basic Python (again please talk to me if this is your first code rodeo!)

Check out Shapely's user manual:

https://shapely.readthedocs.io/en/stable/manual.html#