

Biodiversity Intactness Change in Phoenix (2017–2020)

Peter Vitale

12/6/25

This project uses the Microsoft Planetary Computer STAC catalog, biodiversity raster tiles, and the Phoenix boundary to compute change in BII (Biodiversity Intactness Index) and visualize areas where BII was lost between 2017 and 2020 in Phoenix Arizona.

Import Required Libraries

```
In [1]: import pandas as pd
import geopandas as gpd

import xarray as xr
import rioxarray as rioxr
from shapely.geometry import box

from pystac_client import Client
import planetary_computer

from geogif import gif
import matplotlib.pyplot as plt
import matplotlib.patches as patch
import contextily
```

Connect to Planetary Computer STAC API and Load Catalog

We authenticate and access the STAC service, convert the collection generator to a list, and confirm how many dataset collections exist.

```
In [2]: # Access MPC catalog
catalog = Client.open(
    "https://planetarycomputer.microsoft.com/api/stac/v1",
    modifier=planetary_computer.sign_inplace,
)

catalog.get_collections()
collections = list(catalog.get_collections()) # convert sequence to list for
```

```
# Print the number of collections
print('Number of collections:', len(collections))

# Pull out the NAIP collection
naip_collection = catalog.get_child('naip')
naip_collection
```

Number of collections: 126

```
Out[2]: type "Collection"
id "naip"
stac_version "1.1.0"
description "The [National Agriculture Imagery Program] (https://www.fsa.usda.gov/programs-and-services/aerial-photography/imagery-programs/naip-imagery/) (NAIP) provides U.S.-wide, high-resolution aerial imagery, with four spectral bands (R, G, B, IR). NAIP is administered by the [Aerial Field Photography Office] (https://www.fsa.usda.gov/programs-and-services/aerial-photography/) (AFPO) within the [US Department of Agriculture] (https://www.usda.gov/). Data are captured at least once every three years for each state. This dataset represents NAIP data from 2010-present, in [cloud-optimized GeoTIFF] (https://www.cogeo.org/) format. You can visualize the coverage of current and past collections [here] (https://naip-usdaonline.hub.arcgis.com/)."
▶ links [] 6 items
▶ stac_extensions [] 1 items
▶ item_assets
  msft:region "westeurope"
  msft:container "naip"
  msft:storage_account "naipeuwest"
  msft:short_description "NAIP provides US-wide, high-resolution aerial imagery. This dataset includes NAIP images from 2010 to the present."
  title "NAIP: National Agriculture Imagery Program"
▶ extent
  license "other"
▶ keywords [] 7 items
▶ providers [] 3 items
▶ summaries
▶ assets
```

Locate Available NAIP + Biodiversity Data

`io-biodiversity` is the dataset we search within. We define a bounding box around Phoenix and specify a period window (2017–2020).

```
In [3]: # Temporal range of interest during drought
time_range = '2017-01-01/2020-01-01'
```

```
In [4]: # Bounding box containing phoenix
bbox = [-112.826843,
         32.974108,
         -111.184387,
         33.863574]
```

```
In [5]: search = catalog.search(
    collections = ['io-biodiversity'],
```

```

    bbox = bbox,
    datetime = time_range
)

```

In [6]: # Get items from search

```

items = search.item_collection() # results returned as STAC ItemCollection

# Determine number of items in search
len(items)

```

Out[6]: 4

We need to look at the items to find the years we are interested in

In [7]: items

Out[7]: type "FeatureCollection"
▶ features [] 4 items

We see that 2017 is in item 0 and 2020 is in item 3 now we can

Select and Load Raster Assets

We pull two tiles, one for 2017 and one for 2020, then open them using `rioxarray`.

In [8]: # Get first item in the catalog search

```

item1 = items[0]
item2 = items[3]

```

In [9]: # Examine the item assets to find where the data lies

```

item1.assets

```

Out[9]: {
'data': <Asset href="https://pcdata01euw.blob.core.windows.net/impact/bii-v
1/bii_2020/bii_2020_34.74464974521749_-115.38597824385106_cog.tif?st=2025-1
2-05T21%3A11%3A17Z&se=2025-12-06T21%3A56%3A17Z&sp=rl&sv=2025-07-05&s
r=c&sko
id=9c8ff44a-6a2c-4dfb-b298-1c9212f64d9a&sktid=72f988bf-86f1-41af-91ab-2d7cd
011db47&skt=2025-12-06T05%3A01%3A40Z&ske=2025-12-13T05%3A01%3A40Z&sks=b&
skv
=2025-07-05&sig=P19A3wEQFrFJmalAD7qfohxeQbhiegIBP8EB5ycX0sMc%3D>,
'tilejson': <Asset href="https://planetarycomputer.microsoft.com/api/data/v
1/item/tilejson.json?collection=io-biodiversity&item=bii_2020_34.7446497452
1749_-115.38597824385106_cog&assets=data&tile_format=png&colormap_name=io-b
ii&rescale=0%2C1&expression=0.97%2A%28data_b1%2A2A3.84%29&format=png>,
'rendered_preview': <Asset href="https://planetarycomputer.microsoft.com/ap
i/data/v1/item/preview.png?collection=io-biodiversity&item=bii_2020_34.7446
4974521749_-115.38597824385106_cog&assets=data&tile_format=png&colormap_nam
e=io-bii&rescale=0%2C1&expression=0.97%2A%28data_b1%2A2A3.84%29&format=png
>}

The data is encoded to the 'data' asset so lets call both years as `rioxarrays`

In [10]: phoenix_2020 = rioxr.open_rasterio(item1.assets['data'].href)

```
phoenix_2017 = rioxr.open_rasterio(item2.assets['data'].href)
```

Load Phoenix Boundary Shapefile

We extract the Phoenix municipal boundary from the Arizona shapefile to use for clipping.

```
In [11]: arizona_shape = gpd.read_file('data/tl_2023_04_cousub/tl_2023_04_cousub.shp')
```

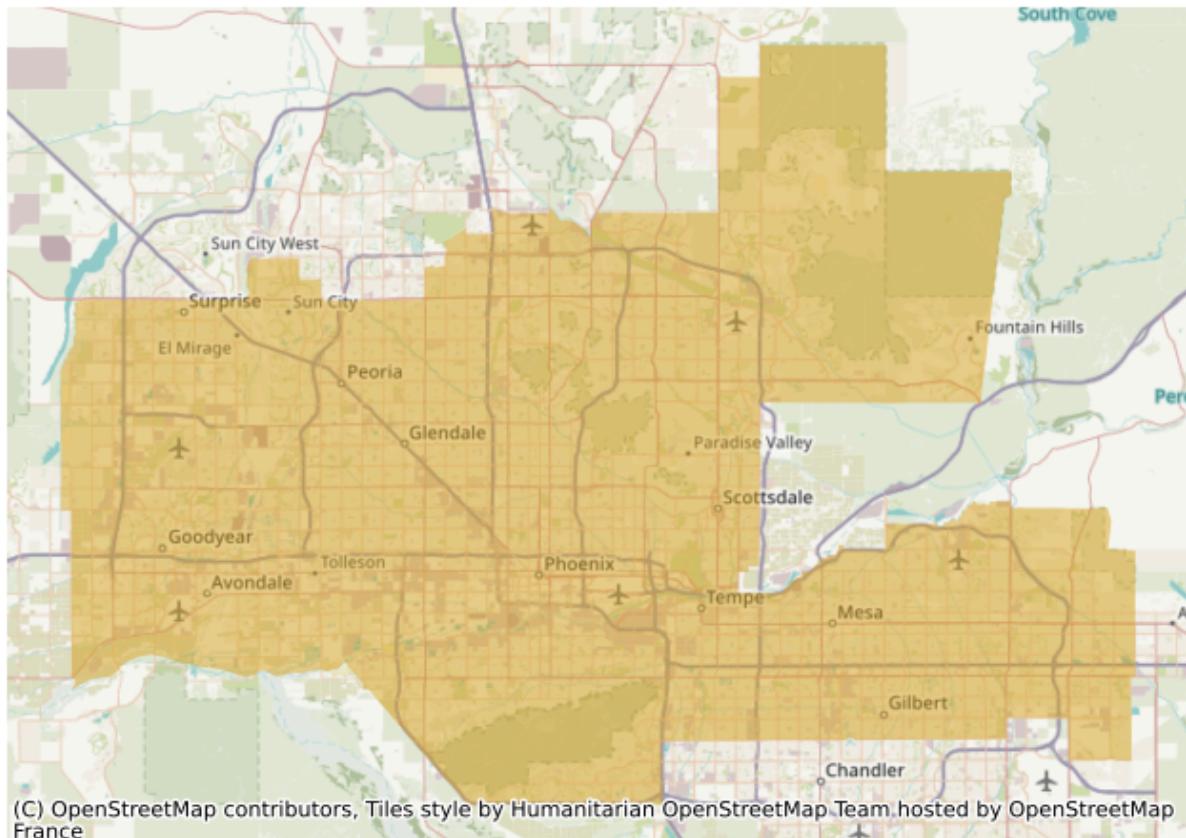
```
In [12]: phoenix_shape = arizona_shape[arizona_shape['NAME'] == 'Phoenix'] # Select c
```

Quick Visual Map Check

Plot boundary with basemap to confirm geometry location

```
In [13]: fig, ax = plt.subplots(figsize=(8, 6))
# Remove the axis for a cleaner map

ax.axis('off')
phoenix_shape.to_crs('EPSG:3857').plot(ax = ax, # Update crs to match contextily
                                         alpha = .5,
                                         color = 'goldenrod')
contextily.add_basemap(ax = ax)
plt.show()
```



Raster Clipping to Phoenix Boundary

We reproject the vector boundary to match raster CRS, then clip both years to the city boundary.

```
In [14]: # Match CRSs
phoenix_match = phoenix_shape.to_crs(phoenix_2020.rio.crs)

# Clip large rasters to Phoenix shape
phoenix_2020 = phoenix_2020.rio.clip_box(*phoenix_match.total_bounds).rio.clip(
    phoenix_2017 = phoenix_2017.rio.clip_box(*phoenix_match.total_bounds).rio.clip(
        # Originally this was two steps, like in class assignments, however that deg
```

Compute BII Threshold (≥ 0.75 Intactness)

We calculate the proportion of pixel area meeting the threshold in 2017 and 2020. The threshold is set as greater than .75 which we consider *high* intactness.

```
In [15]: BII_2017 = phoenix_2017 >= .75
BII_2020 = phoenix_2020 >= .75
```

Calculate percentage of BII in 2017 and 2020

```
In [16]: percentage_2017 = (BII_2017.sum() / BII_2017.count())*100
print(f'In 2017 {round(percentage_2017.item(),2)}% of Phoenix had a BII above .75')
percentage_2020 = (BII_2020.sum() / BII_2020.count())*100
print(f'In 2020 {round(percentage_2020.item(),2)}% of Phoenix had a BII above .75')
```

In 2017 4.18% of Phoenix had a BII above .75
 In 2020 3.81% of Phoenix had a BII above .75

Calculate Biodiversity Loss Areas

Find where 2017 was ≥ 0.75 but dropped below threshold by 2020.

Both BII's are long lists of trues and falses so we find where $\text{BII} \geq .75$ is `True` in 2017 & `False` in 2020

```
In [17]: phoenix_BII_loss = BII_2017 & (~BII_2020)
```

Use `.where` to mask BII loss

`.where` selects based on a condition, and importantly returns every zero as NA

```
In [18]: phoenix_BII_loss_masked = phoenix_BII_loss.where(phoenix_BII_loss!=0) # Select
phoenix_BII_loss_masked
```

```
Out[18]: xarray.DataArray (band: 1, y: 583, x: 990)
```

```
array([[[nan, nan, nan, ..., nan, nan, nan],
       [nan, nan, nan, ..., nan, nan, nan],
       [nan, nan, nan, ..., nan, nan, nan],
       ...,
       [nan, nan, nan, ..., nan, nan, nan],
       [nan, nan, nan, ..., nan, nan, nan],
       [nan, nan, nan, ..., nan, nan, nan]]])
```

▼ Coordinates:

band	(band)	int64 1		
x	(x)	float64 -112.5 -112.5 ... -111.6 -111.6		
y	(y)	float64 33.81 33.81 33.81 ... 33.29 33.29		
spatial_ref	()	int64 0		

► Indexes: (3)

► Attributes: (0)

Final Map — 2020 BII + Lost Areas Overlay

Green background = BII levels in 2020

Red overlay = Areas that were ≥ 0.75 in 2017 but fell below by 2020

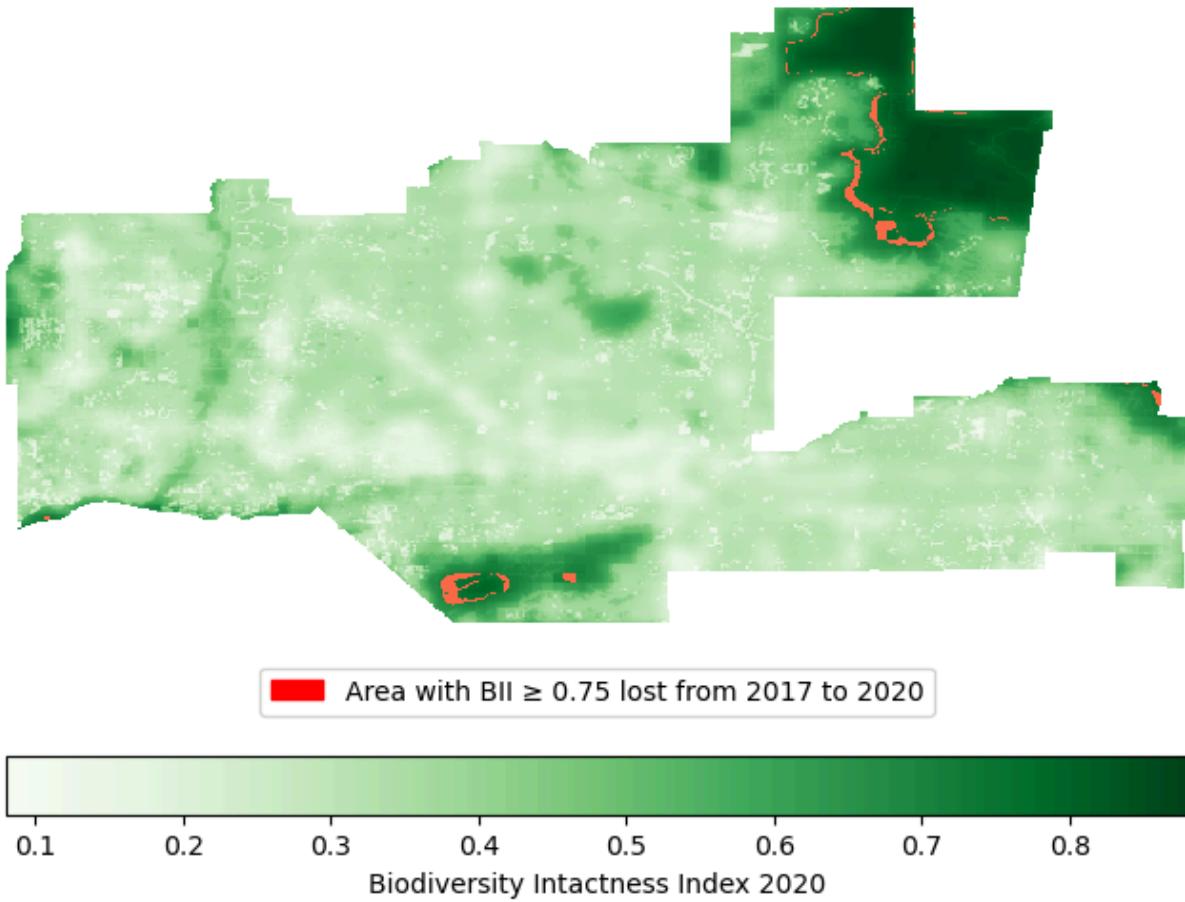
```
In [19]: fig, ax = plt.subplots(figsize=(8, 6))

# Remove the axis for a cleaner map
ax.axis('off')

# Create choropleth map of communities
phoenix_2020.plot(cmap = 'Greens',
                    cbar_kwarg={ 'location': 'bottom', # Edit color bar location
                                'label': 'Biodiversity Intactness Index 2020' })
```

```
phoenix_BII_loss_masked.plot(  
    ax=ax,  
    cmap='Reds',  
    add_colorbar = False) # Remove color bar  
  
# Add title and subtitle for better context  
plt.suptitle('Biodiversity Intactness Index',  
             fontsize=18,  
             weight = 'bold')  
ax.set_title('Phoenix Subdivision',  
            fontsize=14,  
            weight='bold',  
            pad=15)  
  
# Add legend of BII loss as `patch`  
legend_elements = [patch.Patch(facecolor='red', edgecolor='r',  
                                label='Area with BII  $\geq 0.75$  lost from 2017 to 2020')  
  
ax.legend(handles=legend_elements, loc='upper center', bbox_to_anchor=(0.5,  
  
plt.show()
```

Biodiversity Intactness Index Phoenix Subdivision



Map discussion

The map illustrates noticeable reductions in high-intactness habitat (dark green areas, ≥ 0.75 BII) between 2017 and 2020 within the Phoenix region. These shrinking green patches appear mostly around intact habitat cores, suggesting gradual fragmentation of remaining natural areas. This pattern could reflect the combined impacts of urban expansion and ongoing conversion of vegetated land to residential or commercial space.

Visually, many high-intactness pixels align with parks and riparian corridors visible on the basemap, supporting the interpretation that Phoenix's remaining ecological value is concentrated in managed green spaces rather than continuous natural habitat. The red overlay emphasizes where ecological integrity has dropped below high-quality threshold, a pattern consistent with slow but steady urban encroachment.