AnalysisNote6

February 24, 2023

1 Converting HEX GRID

Using cleaned data, we convert GBIF data's GIF on HEX Grid

[1]:
```
import numpy as np
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
import geopandas as gpd
import sys
#import rioxarray as rxr  we used this function to read tiff, but could cause␣
 ↪conflict with plotting hex grids
import h3

#from geojson import Feature, Point, FeatureCollection, Polygon
import plotly.express as px
```

[2]:
```
def AddHexGrid(target):
    meta_gbif=pd.read_csv('./GBIF/For_gbif_trickstar.xlsx - Sheet1.csv') #␣
 ↪which species corresponds which trickster
    extract=meta_gbif[meta_gbif['Category']==target]
    N=len(extract)
    for i in range(N):
        path='./GBIF/'+target+'/'+extract.iloc[i, 2]+'_cleaned.csv'
        df=pd.read_csv(path)
        resolution =1 # We have 842  grids. See https://h3geo.org/docs/
 ↪core-library/restable
        hex_index=[]
        for i in range(len(df)):
            index=h3.geo_to_h3(df.iloc[i, 1], df.iloc[i, 0], resolution)
            hex_index.append(index)
        df['hex_index']=hex_index
        df.to_csv(path, index=False)
```

[59]:
```
AddHexGrid('opposum')
```

[3]:
```
AddHexGrid('ground_squirrel')
```

[60]:
```
AddHexGrid('racoon')
```

[61]:
```
AddHexGrid('mink')
```

[67]:
```
AddHexGrid('mouse')
```

[68]:
```
AddHexGrid('rat')
```

[69]:
```
AddHexGrid('spider')
```

[70]:
```
AddHexGrid('owl')
```

[5]:
```
AddHexGrid('rabbit')
```

[6]:
```
AddHexGrid('hawk')
```

[7]:
```
AddHexGrid('porcupine')
```

[8]:
```
AddHexGrid('anteater')
```

[12]:
```
AddHexGrid('badger')
```

[13]:
```
AddHexGrid('raven')
```

[11]:
```
AddHexGrid('wren')
```

[3]:
```
AddHexGrid('skunk')
```

[4]:
```
AddHexGrid('ground squirrel')
```

2 Comparison of the distributions

In the distributions of real animals, we do not care about the number of observartions per grid. Rather we focus on the presence/absence of species. Below, we will see examples of opposum and racoon, where we have a few data and easily to compare with GBIF (for debug), as well as with the data of tricksters.

It seems that existence of the focal species is necessariy for trickster but not sufficient (although this may be because we have imperfect trickster data).

Note that the problem of boundary on the longtitude would occur. But here we ignore of this plot problem.

[7]:
```
from geojson import Feature, Point, FeatureCollection#, Polygon

def hexagons_dataframe_to_geojson(df_hex, hex_id_field,geometry_field,␣
 ↪value_field,file_output = None):
```

```
    list_features = []

    for i, row in df_hex.iterrows():
        feature = Feature(geometry = row[geometry_field],
                          id = row[hex_id_field],
                          properties = {"value": row[value_field]})
        list_features.append(feature)

    feat_collection = FeatureCollection(list_features)

    if file_output is not None:
        with open(file_output, "w") as f:
            json.dump(feat_collection, f)

    else :
      return feat_collection
```

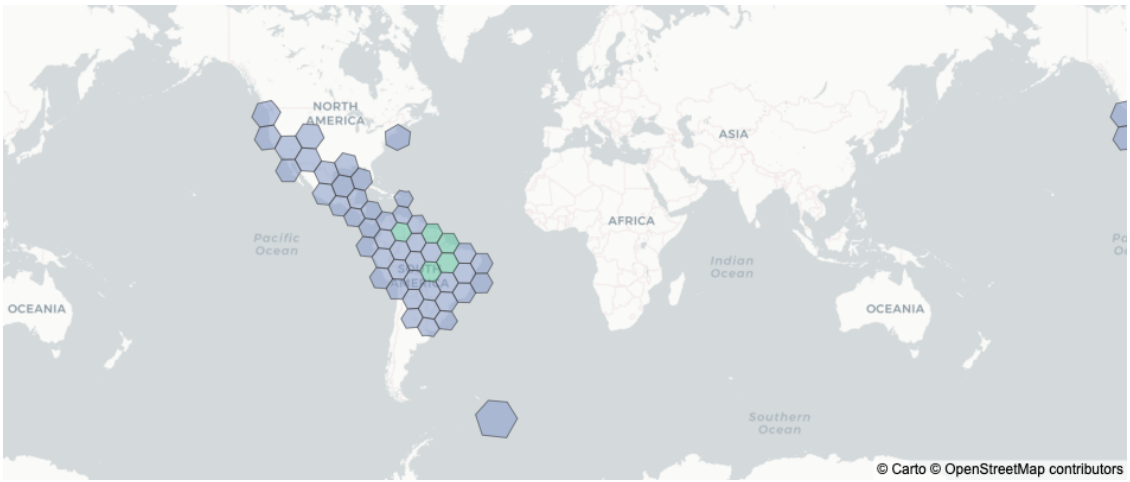
Below we will plot the distributions of rat and mouse as trickster and real animal, respectively

[8]:
```
from shapely.geometry import Polygon

def Double_distributions(target):
    # target: str of  species
    # plot tricksters

    df_TS=pd.read_csv('TrickSter_data3.csv')
    df_TS=df_TS[df_TS['TrickSter']==target]
    """
    df_TS_hex=pd.DataFrame({'hex_index':pd.unique(df_TS['hex_index']),
                            'Presence':np.ones([len(pd.
 ↪unique(df_TS['hex_index']))])})
    Poly=[]
    for i in range (len(df_TS_hex)):
        Poly.append(Polygon(h3.h3_to_geo_boundary(df_TS_hex.iloc[i, 0], True)))
    df_TS_hex['geometry'] = Poly
    geojson_obj = (hexagons_dataframe_to_geojson
                    (df_TS_hex,
                     hex_id_field='hex_index',
                     value_field='Presence',
                     geometry_field='geometry'))
    fig = (px.choropleth_mapbox(
                         df_TS_hex,
                         geojson=geojson_obj,
                         locations='hex_index',
                         color='Presence',
                         color_continuous_scale="Blues",
```

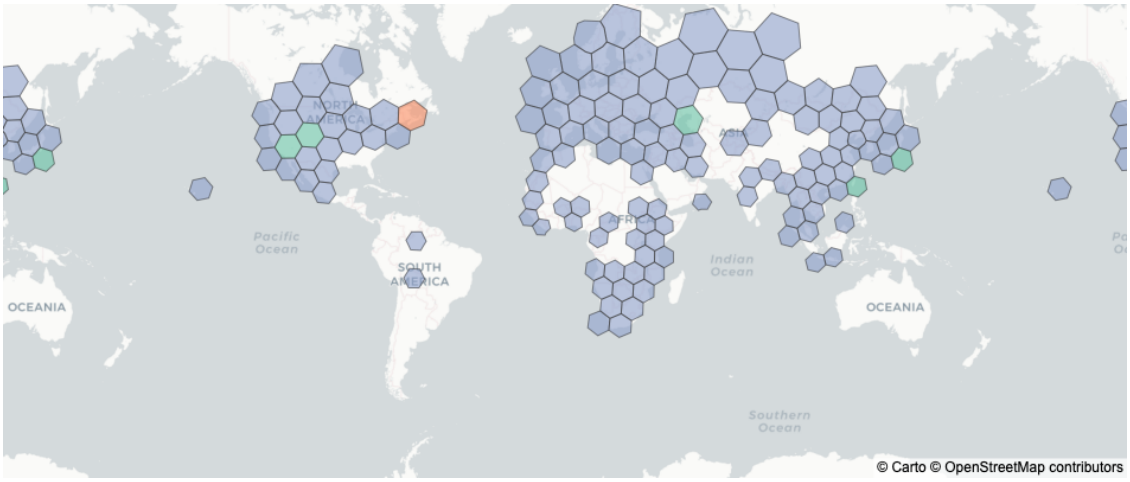
```
                        range_color=(0,1),
                       mapbox_style='carto-positron',
                        zoom=0.75,
                        center = {"lat": 0.0, "lon": 0.0},
                        opacity=1,
                        ))
    fig.update_layout(margin={"r":0,"t":0,"l":0,"b":0})
    #fig.show()
    """

    # plot real animals
    df_meta=pd.read_csv('./GBIF/For_gbif_trickstar.xlsx - Sheet1.csv')
    df_extract=df_meta[df_meta['Category']==target]['Taxa'] # real animal taxa␣
↪corresponding to the target Trickster
    df_extract=df_extract.reset_index(drop=True)
    for j in range(len(df_extract)):
        animal_hex=pd.read_csv('./GBIF/'+target+'/'+df_extract[j]+'_cleaned.
↪csv')['hex_index'].unique()
        if j==0:
            df_animal=animal_hex
        else:
            df_animal=np.concatenate([df_animal, animal_hex], axis=0)
    presence=[]
    TS_index=df_TS['hex_index'].to_list()

    df_animal=np.unique(df_animal)
    for i in range(len(df_animal)):
        if df_animal[i] in TS_index:
            presence.append('Both') # both RA and TS
        else:
            presence.append('Only RA') # only RA
    df_animal=df_animal.tolist()
    for i in range(len(TS_index)):
        if TS_index[i] not in df_animal:
            df_animal.append(TS_index[i])
            presence.append('Only TS') # only TS
    df_animal_hex=pd.DataFrame({'hex_index':df_animal,
                                'Presence':presence})
    Poly=[]
    for i in range (len(df_animal_hex)):
        x=h3.h3_to_geo_boundary(df_animal_hex.iloc[i, 0], True)

        y=np.asanyarray(x)
        if np.any(y[:,0]<-161) or np.any(y[ :, 0]>170):
            for i in range(len(y)):
                if y[i,0]<0:
                    y[i, 0] =360+y[i, 0]
```

```
            x=tuple(map(tuple, y))
        #Poly.append(Polygon(h3.h3_to_geo_boundary(df_animal_hex.iloc[i, 0],␣
↪True)))

        Poly.append(Polygon(x))
    df_animal_hex['geometry'] = Poly
    geojson_obj = (hexagons_dataframe_to_geojson
                   (df_animal_hex,
                    hex_id_field='hex_index',
                    value_field='Presence',
                    geometry_field='geometry'))
    fig = (px.choropleth_mapbox(
                       df_animal_hex,
                       geojson=geojson_obj,
                       locations='hex_index',
                       color='Presence',
                       color_discrete_map={'Both':'#66c2a5','Only RA':
↪'#8da0cb','Only TS':'#fc8d62'},
                       range_color=([0,2]),
                      mapbox_style='carto-positron',
                       zoom=0.5,
                       center = {"lat": 0.0, "lon": 0.0},
                       opacity=0.6,
                       width=800, height=400))
    fig.update_layout(margin={"r":0,"t":0,"l":0,"b":0})
    fig.update_traces(showlegend=False)
    fig.write_image("WorldMap_"+target+".pdf")
    fig.show()
```

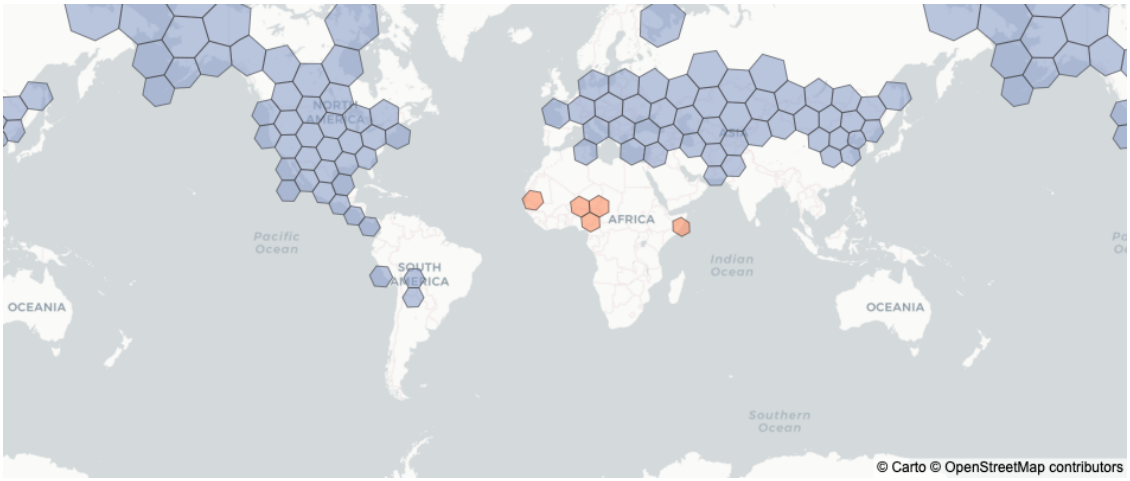
[14]:
```
Double_distributions('anteater')
```



[15]:
```
Double_distributions('badger')
```



[16]:
```
Double_distributions('ground squirrel')
```



[75]:
```
x=h3.h3_to_geo_boundary('81057fffffffffff', True)
y=np.asanyarray(x)
if np.any(y[:,0]<-161) or np.any(y[ :, 0]>170):
    for i in range(len(y)):
        if y[i,0]<0:
            y[i, 0] =360+y[i, 0]
y
```

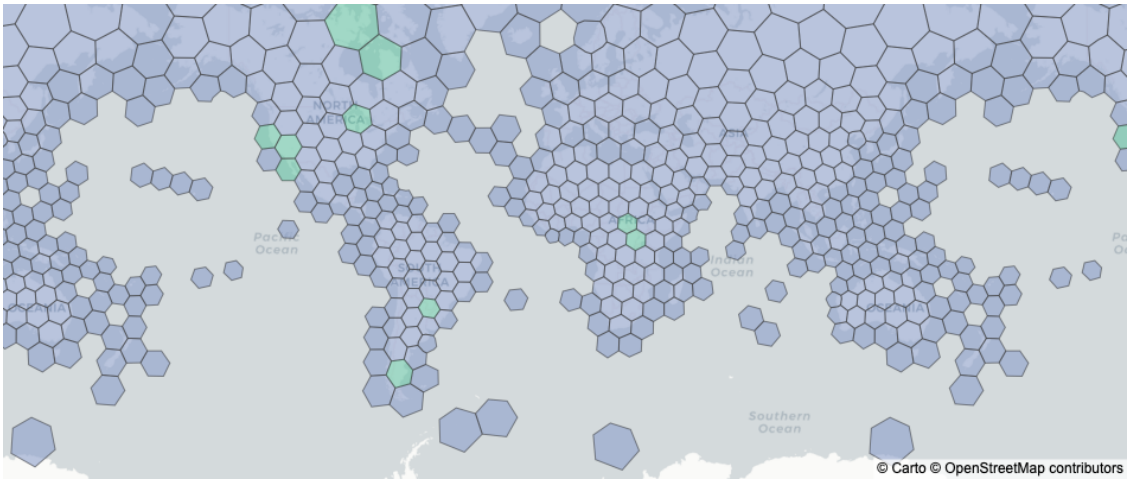
[75]:
```
array([[118.57660604,  83.15405442],
       [136.93797037,  79.32046591],
       [160.68630381,  78.12011724],
```

```
        [183.63273388,  79.98369866],
        [198.10828199,  84.22628942],
        [145.55819769,  87.36469532],
        [118.57660604,  83.15405442]])
```

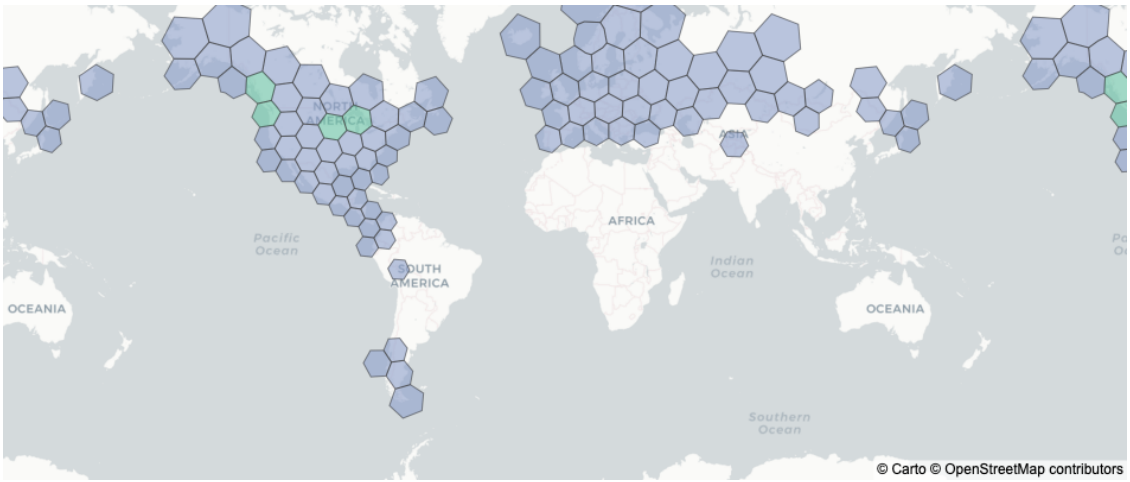
[76]:
```
x=h3.h3_to_geo_boundary('810dbffffffffff')
y=np.asanyarray(x)
y[:, 0]<-160
y[:,0]+360
```

[76]:
```
array([430.79472682, 426.19292316, 423.29242333, 424.21420041,
       428.21921735, 431.97154653])
```

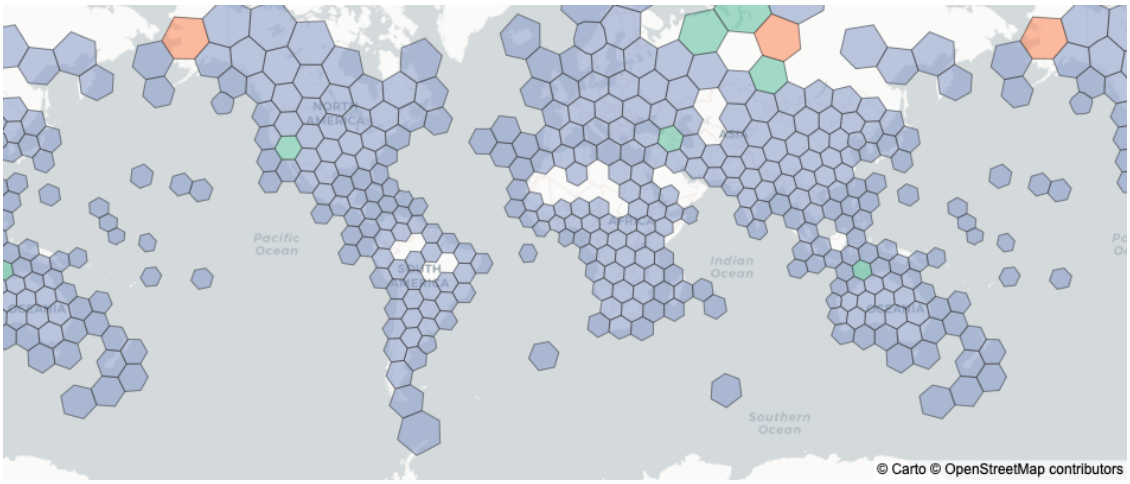
[17]:
```
Double_distributions('hawk')
```



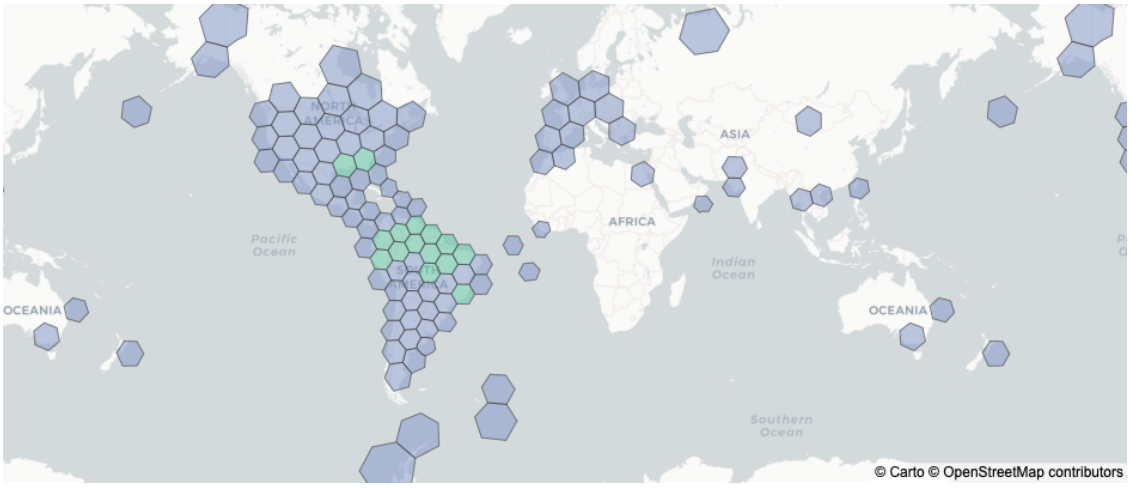
[18]:
```
Double_distributions('mink')
```



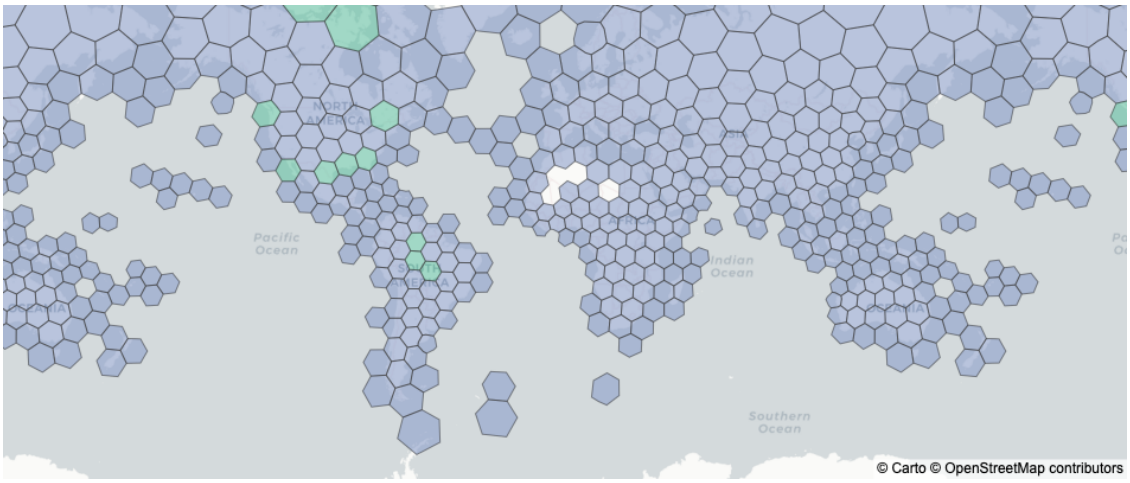
[19]:
```
Double_distributions('mouse')
```



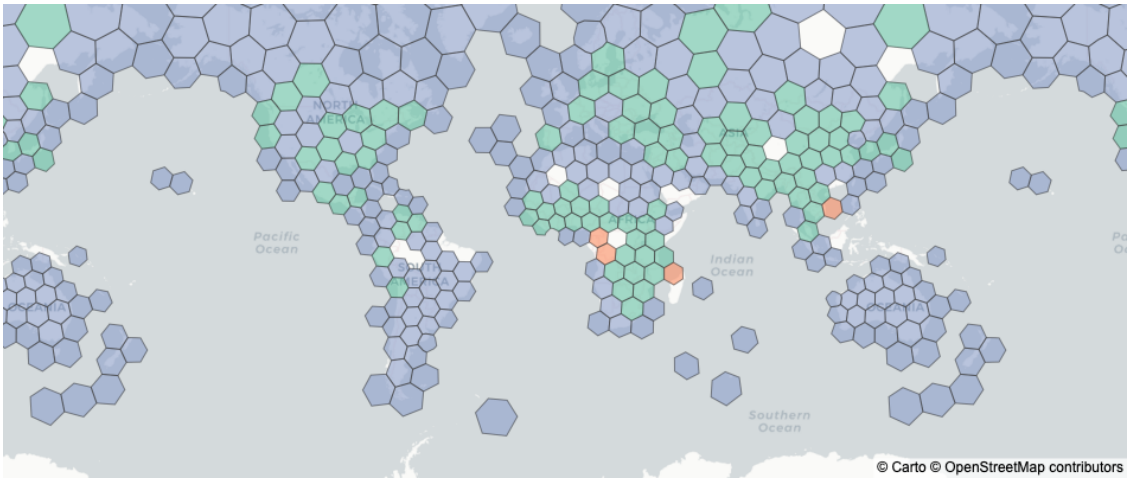
[20]:
```
Double_distributions('opossum')
```



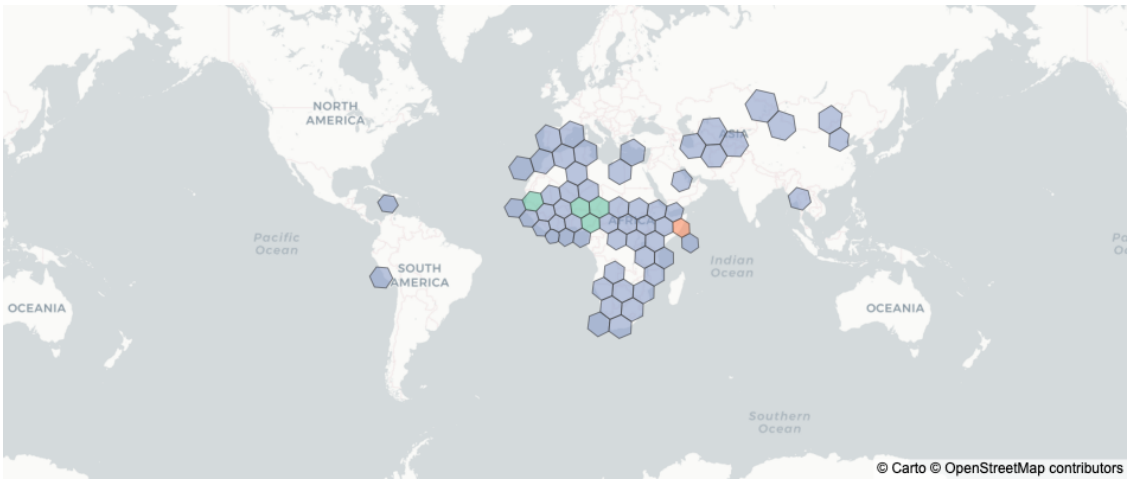
[21]:
```
Double_distributions('owl')
```



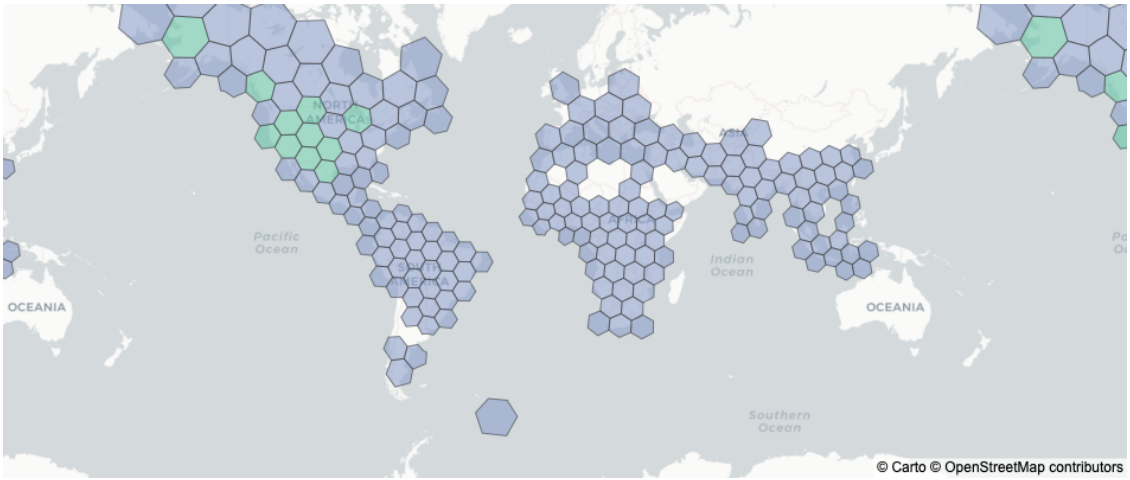
[23]:
```
Double_distributions('rabbit')
```



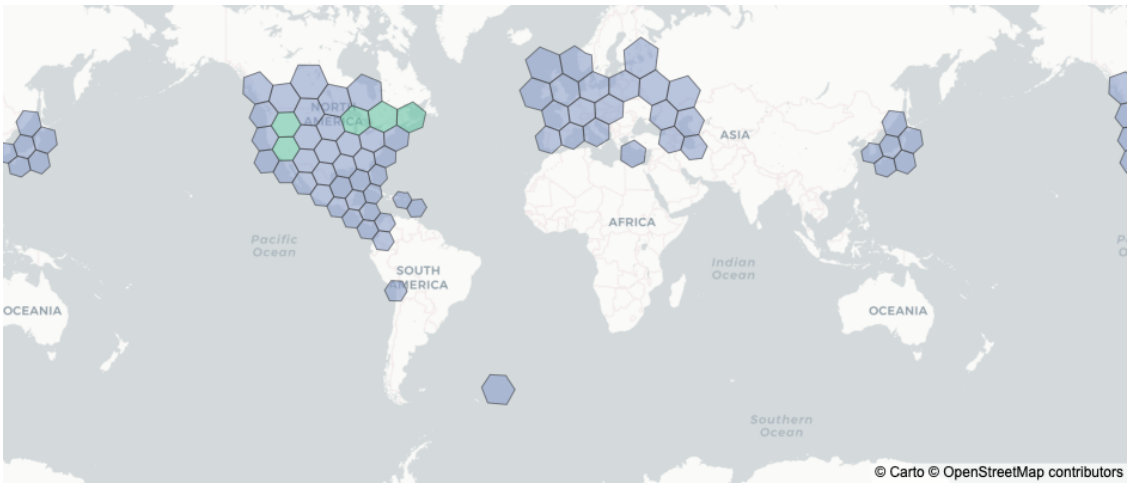
[9]:
```
Double_distributions('ground squirrel')
```



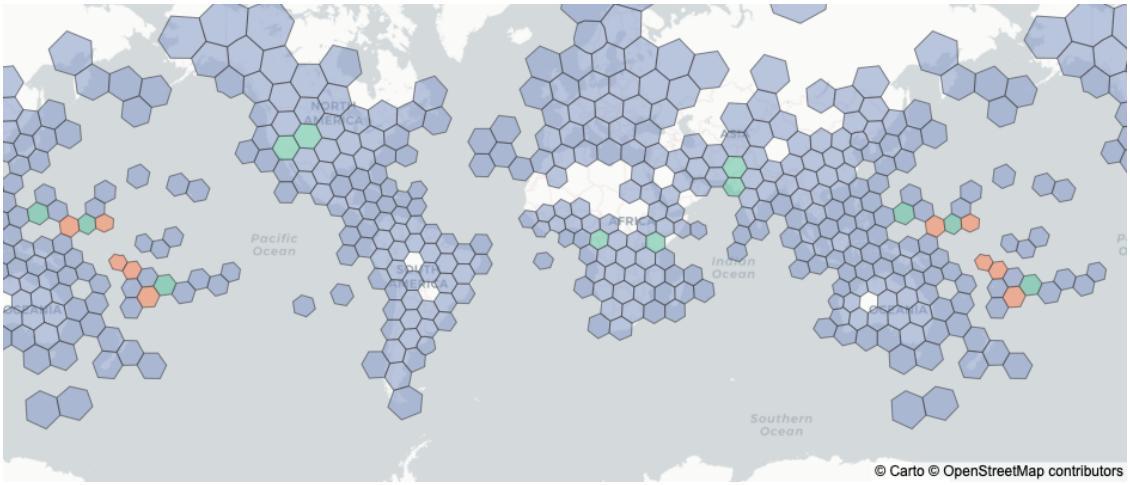
[22]:
```
Double_distributions('porcupine')
```



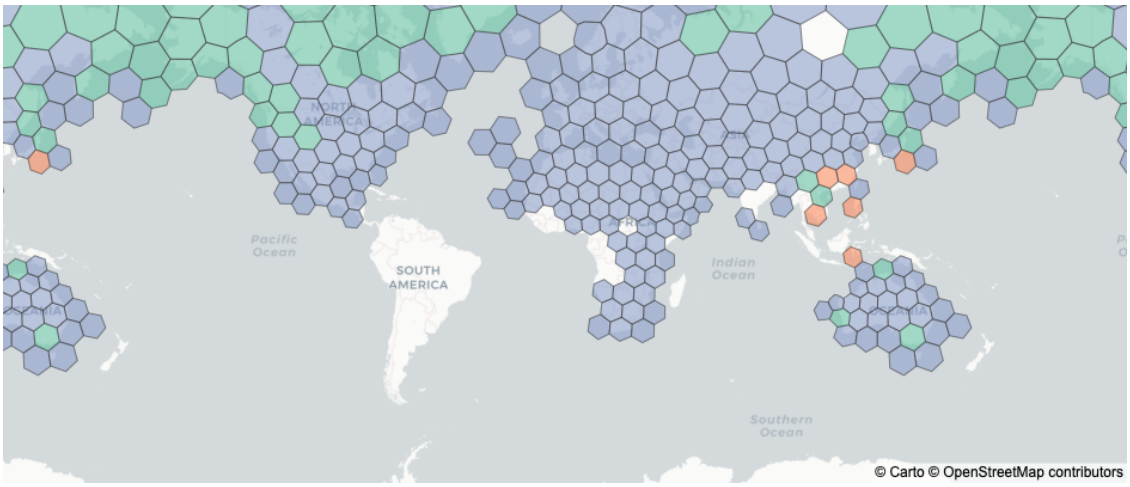
[24]:
```
Double_distributions('raccoon')
```



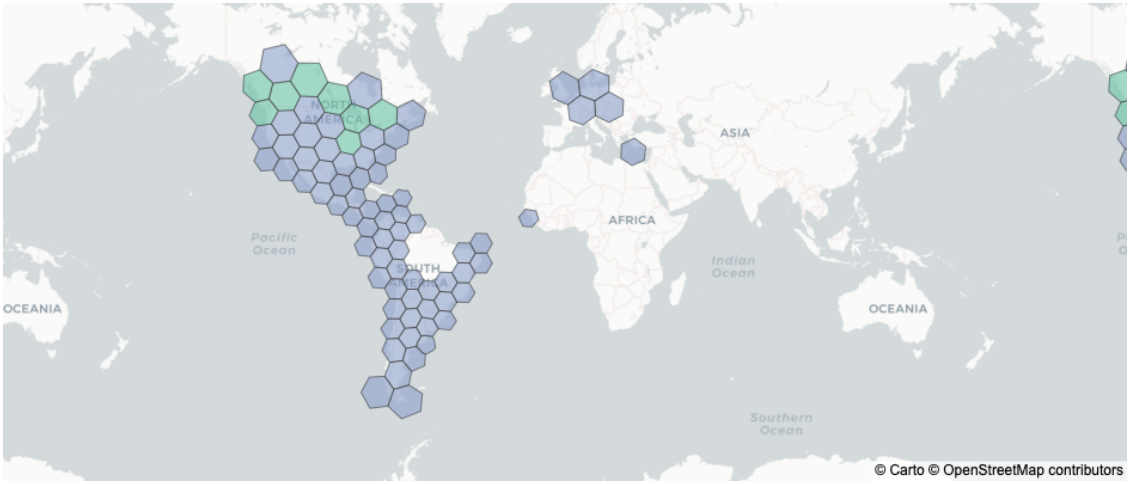
[25]:
```
Double_distributions('rat')
```



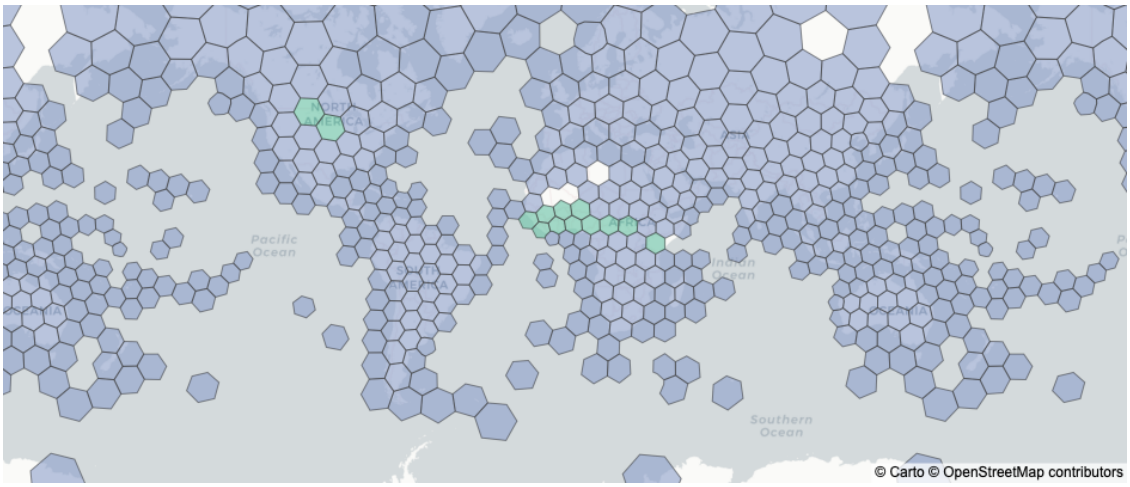
[26]:
```
Double_distributions('raven')
```



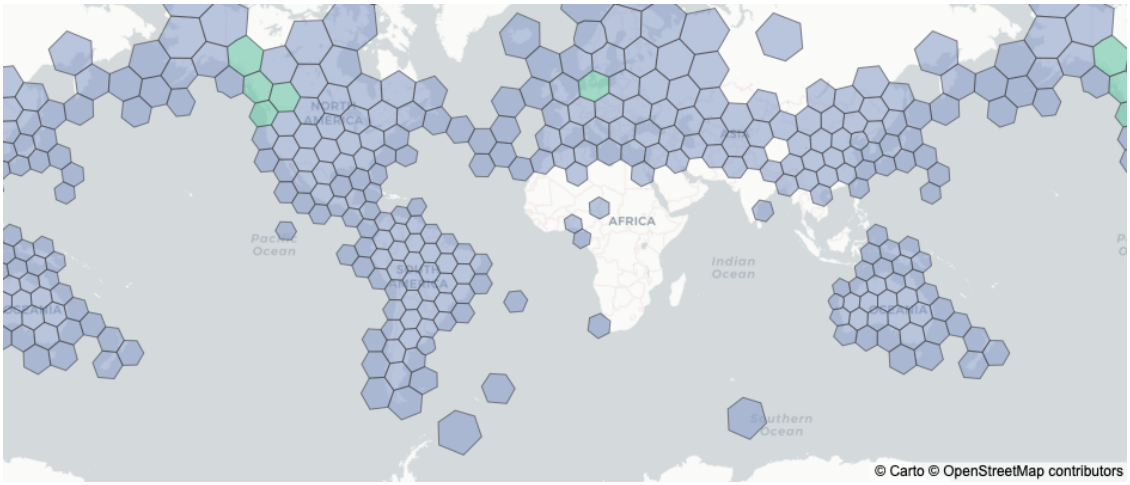
[27]:
```
Double_distributions('skunk')
```



[28]:
```
Double_distributions('spider')
```



[29]:
```
Double_distributions('wren')
```



[ ]:
```

```