Objective

To produce a visualization that effectively tells the story of variations of tortoise migration patterns.

Sub-objectives

1. Create plots to show whether and how Alison and Christian differ their migration patterns over the years.
2. Design and produce plot(s) that effectively tell the story of multiple tortoises in a single plot.

The Data

In [1]:
```
import pandas as pd
import numpy as np
import pandas as pd
import geopandas as gpd
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import datetime, timedelta
import folium

import skmob
from skmob.preprocessing import compression, detection, clustering

pd.set_option('display.max_columns', 500)
pd.set_option('display.max_rows', 200)
import warnings
warnings.simplefilter("ignore")
```

In [2]:
```
events = pd.read_csv("GalapagostortoiseMovementEcology Programme_2009_2018.csv
new_cols = pd.Series(events.columns.values).str.replace(pat = r"[-:]", repl="_
tortoise = events.set_axis(labels = new_cols, axis =1)
tortoise.drop(tortoise.index[tortoise["manually_marked_outlier"].notna()], inp
tortoise.drop(columns =["manually_marked_outlier"], inplace=True)
tortoise.drop(index = tortoise.index[tortoise.isna().any(axis=1)], inplace=Tru
cols = ['event_id', 'timestamp', 'location_long', 'location_lat',
       'eobs_temperature', 'ground_speed', 'heading', 'height_above_ellipsoid'
       #'individual_taxon_canonical_name', 'tag_local_identifier',
       'individual_local_identifier']
tortoise = tortoise[cols].assign()
tortoise.insert(loc = 2, column = "timestamp_UTC",
    value = tortoise['timestamp'].apply(lambda x: x.tz_localize(tz='UTC'))
)
tortoise.insert(loc = 3, column = "timestamp_local",
    value = tortoise["timestamp_UTC"].apply(lambda x: x.tz_convert(tz='US/Paci
)

tortoise.sort_values(by = ["individual_local_identifier", "timestamp_local"],

tortoise.insert(loc = 4, column = "minute_diff",
    value = tortoise.groupby(['individual_local_identifier'])["timestamp_local
        .apply(lambda x: x/np.timedelta64(1, 'm')).fillna(0).astype('int64')
)

tortoise['year'] = tortoise["timestamp_local"].dt.year
tortoise['month'] = tortoise["timestamp_local"].dt.month
tortoise['date'] = tortoise["timestamp_local"].dt.date
```

In [3]:
```
from skmob.tessellation import tilers
from skmob.preprocessing import filtering
tessellation = tilers.tiler.get("h3_tessellation", base_shape="isla santa cruz
tessellation["tile_lng"] = tessellation.geometry.centroid.x
tessellation["tile_lat"] = tessellation.geometry.centroid.y
```

For this project, I have focused mainly on the following:

- The events themselves
- The time of the events
- Latitude and longidute (they are how I measured movement across days)
- Individual local identifier (tracking specific tortoises)

I did not use speed, temperature, or heading for my visualizations.

In [4]:
```
tortoise.head()
```

Out[4]:

	event_id	timestamp	timestamp_UTC	timestamp_local	minute_diff	location_lo
1290187	46119277	2010-09-17 17:01:25.998	2010-09-17 17:01:25.998000+00:00	2010-09-17 10:01:25.998000-07:00	0	-90.2418
1290188	46119278	2010-09-17 18:00:58.001	2010-09-17 18:00:58.001000+00:00	2010-09-17 11:00:58.001000-07:00	59	-90.2422
1290189	46119279	2010-09-17 19:00:56.001	2010-09-17 19:00:56.001000+00:00	2010-09-17 12:00:56.001000-07:00	59	-90.2422
1290190	46119280	2010-09-17 20:00:29.000	2010-09-17 20:00:29+00:00	2010-09-17 13:00:29-07:00	59	-90.2422
1290191	46119281	2010-09-17 21:00:56.000	2010-09-17 21:00:56+00:00	2010-09-17 14:00:56-07:00	60	-90.2421

In [5]:
```
# Add columns for year and month respectively so that it is easier to manipula
tortoise['date_year'] = pd.DatetimeIndex(tortoise['date']).year
tortoise['date_month'] = pd.DatetimeIndex(tortoise['date']).month
tortoise.head()
```

Out[5]:

	event_id	timestamp	timestamp_UTC	timestamp_local	minute_diff	location_lo
1290187	46119277	2010-09-17 17:01:25.998	2010-09-17 17:01:25.998000+00:00	2010-09-17 10:01:25.998000-07:00	0	-90.2418
1290188	46119278	2010-09-17 18:00:58.001	2010-09-17 18:00:58.001000+00:00	2010-09-17 11:00:58.001000-07:00	59	-90.2422
1290189	46119279	2010-09-17 19:00:56.001	2010-09-17 19:00:56.001000+00:00	2010-09-17 12:00:56.001000-07:00	59	-90.2422
1290190	46119280	2010-09-17 20:00:29.000	2010-09-17 20:00:29+00:00	2010-09-17 13:00:29-07:00	59	-90.2422
1290191	46119281	2010-09-17 21:00:56.000	2010-09-17 21:00:56+00:00	2010-09-17 14:00:56-07:00	60	-90.2421

In [6]:
```
df = tortoise['date_month'].nunique()
```

Setting Up the Dataframes for Visualizations

In [7]:
```
#from sklearn.metrics.pairwise import haversine_distances
from math import radians
p1 = [-0.6357607,   -90.2440642]
p2 = [-0.6350828,   -90.2435916]
p1_in_radians = [radians(_) for _ in p1]
p2_in_radians = [radians(_) for _ in p2]
#result = haversine_distances([p1_in_radians, p2_in_radians])
#(result * 6371000/1000)[0][1]  # multiply by Earth radius to get kilometers
```

In [8]:
```
p1, p2, p1_in_radians, p2_in_radians
```

Out[8]:
```
([-0.6357607, -90.2440642],
 [-0.6350828, -90.2435916],
 [-0.011096117469783912, -1.5750560506711424],
 [-0.011084285882784643, -1.5750478022450976])
```

In [9]:
```
tortoise_trajectory = skmob.TrajDataFrame(tortoise, latitude='location_lat', l
    user_id='individual_local_identifier', datetime='timestamp_local', \
)
```

Adding libraries to compute radius of gyration. This is one of the methods that is used to track tortoise movement.

In [10]:
```
from skmob.measures.individual import radius_of_gyration
tortoise_radius_gyration = radius_of_gyration(tortoise_trajectory)
tortoise_radius_gyration.head()
```

```
100%|██████████████████████████████████████████
██████████| 96/96 [00:06<00:00, 15.89it/s]
```

Out[10]:

	uid	radius_of_gyration
0	Alison	1.498891
1	Andrea	2.287264
2	Andres	0.036833
3	Anne	1.506432
4	Bill	0.257950

In [11]:
```
alison = tortoise[tortoise["individual_local_identifier"] == "Alison"]
```

In [12]:
```
christian = tortoise[tortoise["individual_local_identifier"] == "Christian"]
```

In [13]:
```
#For looking specifically at Alison's trajectory
alison_trajectory = tortoise_trajectory.query("uid == 'Alison'")
```

In [14]:
```
#For looking specifically at Christian's trajectory
christian_trajectory = tortoise_trajectory.query("uid == 'Christian'")
```

In [15]:
```
#Creating a dataframe that adds the two previous dataframes together so that i
#coordinates for both are visible
al_ch = alison.append(christian)
```

In [16]:
```
al_ch.head()
```

Out[16]:

	event_id	timestamp	timestamp_UTC	timestamp_local	minute_diff	location_lo
1290187	46119277	2010-09-17 17:01:25.998	2010-09-17 17:01:25.998000+00:00	2010-09-17 10:01:25.998000-07:00	0	-90.2418
1290188	46119278	2010-09-17 18:00:58.001	2010-09-17 18:00:58.001000+00:00	2010-09-17 11:00:58.001000-07:00	59	-90.2422
1290189	46119279	2010-09-17 19:00:56.001	2010-09-17 19:00:56.001000+00:00	2010-09-17 12:00:56.001000-07:00	59	-90.2422
1290190	46119280	2010-09-17 20:00:29.000	2010-09-17 20:00:29+00:00	2010-09-17 13:00:29-07:00	59	-90.2422
1290191	46119281	2010-09-17 21:00:56.000	2010-09-17 21:00:56+00:00	2010-09-17 14:00:56-07:00	60	-90.2421

In [17]:
```
radius_of_gyration(alison_trajectory[alison_trajectory["date"] == pd.to_dateti
```

```
100%|██████████| 1/1 [00:00<00:00, 143.22it/s]
```

Out[17]: 0.01951637960295725

In [18]:
```
radius_of_gyration(christian_trajectory[christian_trajectory["date"] == pd.to_
```

```
100%|██████████| 1/1 [00:00<00:00, 139.24it/s]
```

Out[18]: 0.038658831930201296

In [19]:
```
from skmob.measures.individual import radius_of_gyration
alison_rog = []
date_list_a = []
for date in alison.date.unique():
    date_list_a.append(date)
    rog = radius_of_gyration(alison_trajectory[alison_trajectory["date"] == da
    alison_rog.append(rog)
```

In [20]:
```
christian_rog = []
date_list_ch = []
for date in christian.date.unique():
    date_list_ch.append(date)
    rog = radius_of_gyration(christian_trajectory[christian_trajectory["date"]
    christian_rog.append(rog)
```

In [21]:
```
alison_radius_gyration = pd.DataFrame(data = {"date": date_list_a, "rog":alisc
```

In [22]:
```
#Merging alison and alison_radius_gyration dataframes so that alison's rogs ar
#This process is repeated for Christian
alison_n = pd.merge(alison, alison_radius_gyration, on=['date'])
```

In [23]:
```
alison_n.head()
```

Out[23]:

	event_id	timestamp	timestamp_UTC	timestamp_local	minute_diff	location_long	loc
0	46119277	2010-09-17 17:01:25.998	2010-09-17 17:01:25.998000+00:00	2010-09-17 10:01:25.998000-07:00	0	-90.241889	-
1	46119278	2010-09-17 18:00:58.001	2010-09-17 18:00:58.001000+00:00	2010-09-17 11:00:58.001000-07:00	59	-90.242230	-
2	46119279	2010-09-17 19:00:56.001	2010-09-17 19:00:56.001000+00:00	2010-09-17 12:00:56.001000-07:00	59	-90.242210	-
3	46119280	2010-09-17 20:00:29.000	2010-09-17 20:00:29+00:00	2010-09-17 13:00:29-07:00	59	-90.242227	-
4	46119281	2010-09-17 21:00:56.000	2010-09-17 21:00:56+00:00	2010-09-17 14:00:56-07:00	60	-90.242101	-

In [24]:
```
#This is the setup for looking at migration patterns across separate years. It
# Alison and Christian together
alison_n.date_year.unique()
```

Out[24]:
```
array([2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018], dtype=int64)
```

In [25]:
```
alison_n_2010 = alison_n[alison_n['date_year']==2010]
alison_n_2011 = alison_n[alison_n['date_year']==2011]
alison_n_2012 = alison_n[alison_n['date_year']==2012]
alison_n_2013 = alison_n[alison_n['date_year']==2013]
alison_n_2014 = alison_n[alison_n['date_year']==2014]
alison_n_2015 = alison_n[alison_n['date_year']==2015]
alison_n_2016 = alison_n[alison_n['date_year']==2016]
alison_n_2017 = alison_n[alison_n['date_year']==2017]
alison_n_2018 = alison_n[alison_n['date_year']==2018]
```

In [26]:
```
alison_n_years = [alison_n_2010, alison_n_2011, alison_n_2012,
                  alison_n_2013, alison_n_2014, alison_n_2015,
                  alison_n_2016, alison_n_2017, alison_n_2018]
```

In [27]:
```
#Looking at only the top rogs for Alison. This is repeated for Christian and t
alison_top_rogs = alison_n[alison_n['rog']>=alison_n.rog.quantile(.98)]
alison_top_rogs.shape
```

Out[27]: (809, 18)

In [28]:
```
christian_radius_gyration = pd.DataFrame(data = {"date": date_list_ch, "rog":c
```

In [29]:
```
christian_n = pd.merge(christian, christian_radius_gyration, on=['date'])
```

In [30]:
```
christian_n.head()
```

Out[30]:

	event_id	timestamp	timestamp_UTC	timestamp_local	minute_diff	location_long	loc
0	33691506	2010-09-24 13:01:20.000	2010-09-24 13:01:20+00:00	2010-09-24 06:01:20-07:00	0	-91.092650	
1	33691507	2010-09-24 14:00:21.998	2010-09-24 14:00:21.998000+00:00	2010-09-24 07:00:21.998000-07:00	59	-91.092997	
2	77563565	2010-09-24 15:00:49.998	2010-09-24 15:00:49.998000+00:00	2010-09-24 08:00:49.998000-07:00	60	-91.093024	
3	77563566	2010-09-24 16:00:31.001	2010-09-24 16:00:31.001000+00:00	2010-09-24 09:00:31.001000-07:00	59	-91.092979	
4	77563567	2010-09-24 17:00:55.998	2010-09-24 17:00:55.998000+00:00	2010-09-24 10:00:55.998000-07:00	60	-91.092383	

In [31]:
```
christian_top_rogs = christian_n[christian_n['rog']>=christian_n.rog.quantile(
christian_top_rogs.shape
```

Out[31]: (852, 18)

In [32]:
```
christian_n_2010 = christian_n[christian_n['date_year']==2010]
christian_n_2011 = christian_n[christian_n['date_year']==2011]
christian_n_2012 = christian_n[christian_n['date_year']==2012]
christian_n_2013 = christian_n[christian_n['date_year']==2013]
christian_n_2014 = christian_n[christian_n['date_year']==2014]
christian_n_2015 = christian_n[christian_n['date_year']==2015]
christian_n_2016 = christian_n[christian_n['date_year']==2016]
christian_n_2017 = christian_n[christian_n['date_year']==2017]
christian_n_2018 = christian_n[christian_n['date_year']==2018]
```

In [33]:
```
christian_n_years = [christian_n_2010, christian_n_2011, christian_n_2012,
                     christian_n_2013, christian_n_2014, christian_n_2015,
                     christian_n_2016, christian_n_2017, christian_n_2018]
```

In [34]:
```
al_ch_n = alison_n.append(christian_n)
```

In [35]:
```
al_ch_n.head()
```

Out[35]:

	event_id	timestamp	timestamp_UTC	timestamp_local	minute_diff	location_long	loc
0	46119277	2010-09-17 17:01:25.998	2010-09-17 17:01:25.998000+00:00	2010-09-17 10:01:25.998000-07:00	0	-90.241889	-
1	46119278	2010-09-17 18:00:58.001	2010-09-17 18:00:58.001000+00:00	2010-09-17 11:00:58.001000-07:00	59	-90.242230	-
2	46119279	2010-09-17 19:00:56.001	2010-09-17 19:00:56.001000+00:00	2010-09-17 12:00:56.001000-07:00	59	-90.242210	-
3	46119280	2010-09-17 20:00:29.000	2010-09-17 20:00:29+00:00	2010-09-17 13:00:29-07:00	59	-90.242227	-
4	46119281	2010-09-17 21:00:56.000	2010-09-17 21:00:56+00:00	2010-09-17 14:00:56-07:00	60	-90.242101	-

In [36]:
```
al_ch_n_2010 = al_ch_n[al_ch_n['date_year']==2010]
al_ch_n_2011 = al_ch_n[al_ch_n['date_year']==2011]
al_ch_n_2012 = al_ch_n[al_ch_n['date_year']==2012]
al_ch_n_2013 = al_ch_n[al_ch_n['date_year']==2013]
al_ch_n_2014 = al_ch_n[al_ch_n['date_year']==2014]
al_ch_n_2015 = al_ch_n[al_ch_n['date_year']==2015]
al_ch_n_2016 = al_ch_n[al_ch_n['date_year']==2016]
al_ch_n_2017 = al_ch_n[al_ch_n['date_year']==2017]
al_ch_n_2018 = al_ch_n[al_ch_n['date_year']==2018]
```

In [37]:
```
al_ch_n_years = [al_ch_n_2010, al_ch_n_2011, al_ch_n_2012,
                 al_ch_n_2013, al_ch_n_2014, al_ch_n_2015,
                 al_ch_n_2016, al_ch_n_2017, al_ch_n_2018]
```

In [38]:
```
al_ch_n_top_rogs = al_ch_n[al_ch_n['rog']>=al_ch_n.rog.quantile(.98)]
al_ch_n_top_rogs.shape
```

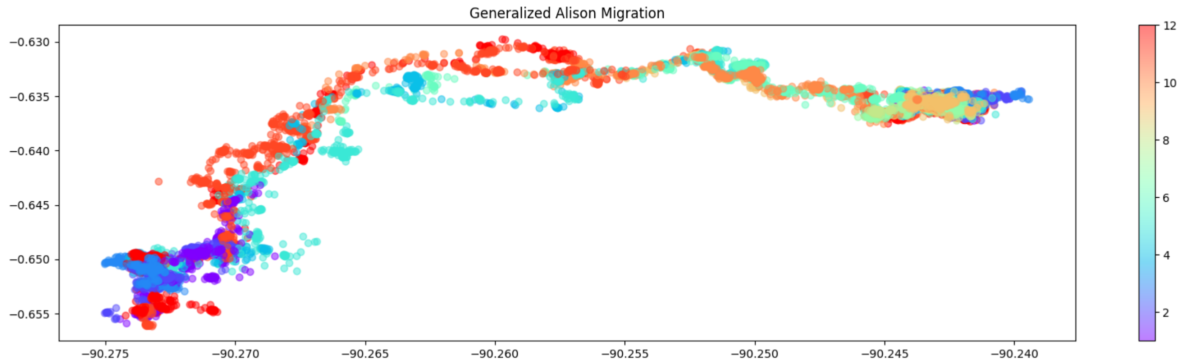
Out[38]:
```
(1661, 18)
```

The Graphs

Alison

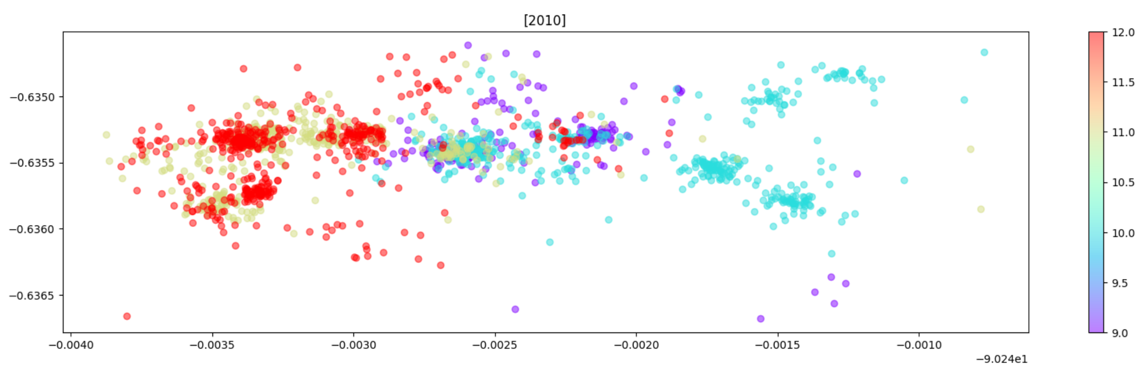
This first graph is to track the overall migration pattern for each month (1 to 12) over all the years of data. As you can see, there is a general pattern between the clusters of points in the bottom left and top right.

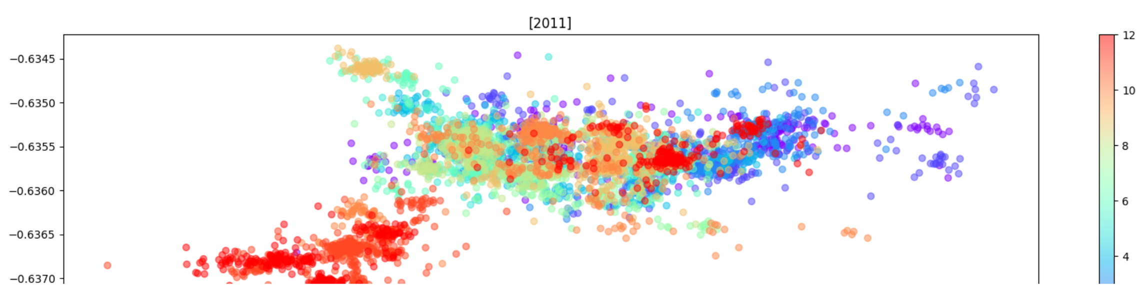
In [39]:
```
plt.figure(figsize = (20,5))
ax = plt.axes()
plt.scatter(alison.location_long.values, alison.location_lat.values,c=alison['
plt.title('Generalized Alison Migration')
plt.colorbar()
plt.show()
```



Because the first plot only gives a general overview of all the years, it was important to look at the migration pattern of each year individually. It is possible to see each year by scrolling through the graphs below.

In [40]:
```
for x in alison_n_years:
    plt.figure(figsize = (20,5))
    ax = plt.axes()
    plt.scatter(x.location_long.values, x.location_lat.values,c=x['date_month'
    plt.title(x.date_year.unique())
    plt.colorbar()
```

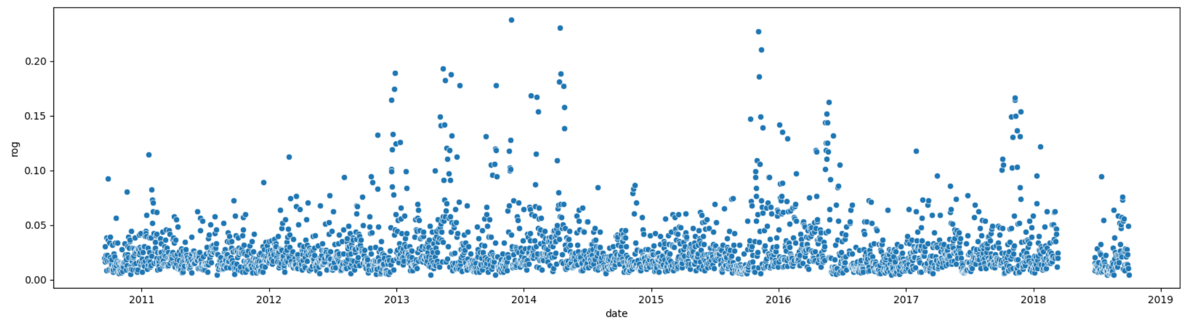




- It is evident that there may have been some tracking issues in 2010 and 2011, because it does not seem consistent at all where Alison traveled. However, in 2012, her migration patterns became significantly more standardized.

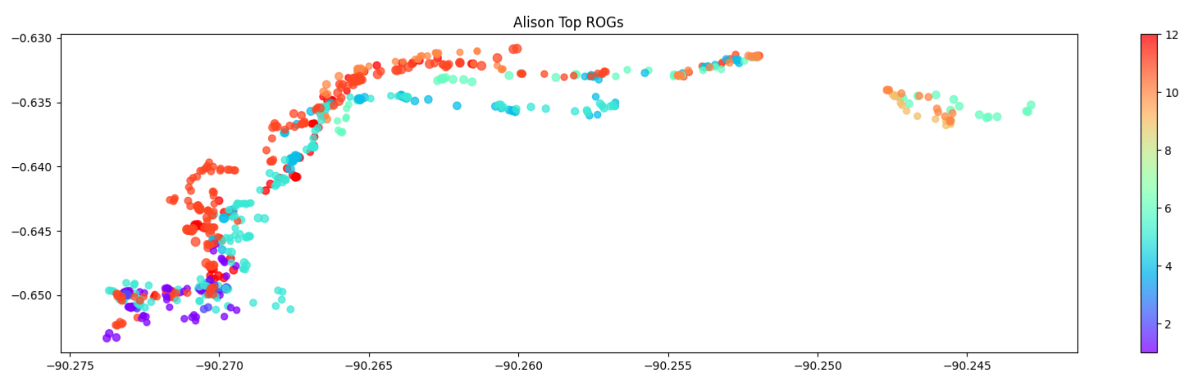
In [41]:
```
#Looking at Alison's rogs. Process is repeated for Christian.
plt.figure(figsize = (20,5))
sns.scatterplot(data = alison_radius_gyration, x="date", y="rog")
```

Out[41]: <Axes: xlabel='date', ylabel='rog'>



In [42]:
```
#Point size corresponds to the size of the rogs
#Rogs multiplied by 250 so that they are easier to see on the graph
plt.figure(figsize = (20,5))
ax = plt.axes()
plt.scatter(alison_top_rogs.location_long.values, alison_top_rogs.location_lat
plt.title('Alison Top ROGs')
plt.colorbar()
```

Out[42]: <matplotlib.colorbar.Colorbar at 0x1a9b3c460e0>



- When looking at this graph, it is evident that there are significantly less points in the bottom left and the top right than there are in Alison's generalized migration graph. This is an indicator that those two clusters are the migration destinations.

Additionally, I wanted to see where Alison traveled on a geographic map for several years of her migration. I did the same for Christian as well.

In [43]:
```
#Maybe change the colors
map_f = alison_trajectory.query("year == 2012").plot_trajectory(zoom=14, start
alison_trajectory.query("year == 2013").plot_trajectory(map_f=map_f, start_enc
alison_trajectory.query("year == 2014").plot_trajectory(map_f=map_f, start_enc
alison_trajectory.query("year == 2015").plot_trajectory(map_f=map_f, start_enc
alison_trajectory.query("year == 2016").plot_trajectory(map_f=map_f, start_enc
map_f
```

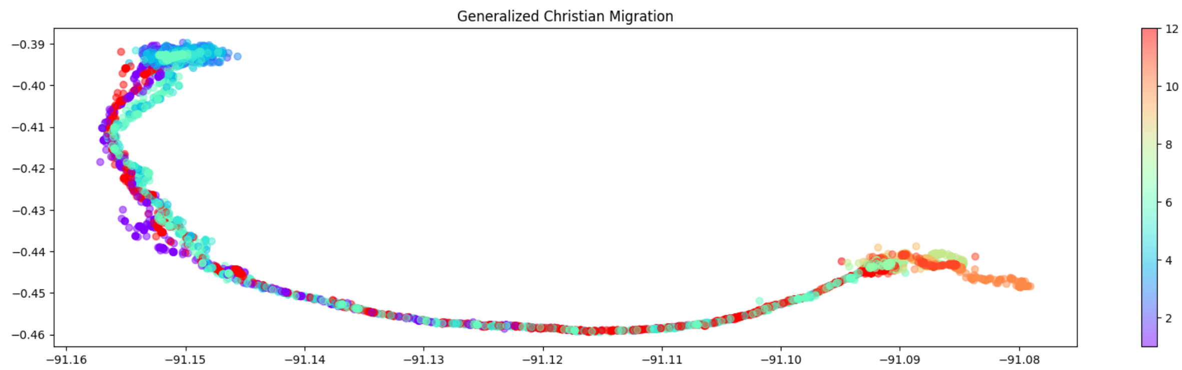
Out[43]: Make this Notebook Trusted to load map: File -> Trust Notebook

Christian

To start, a generalized migration map for Christian was created (just like for Alison). This will be repeated when looking at the two at the same time.

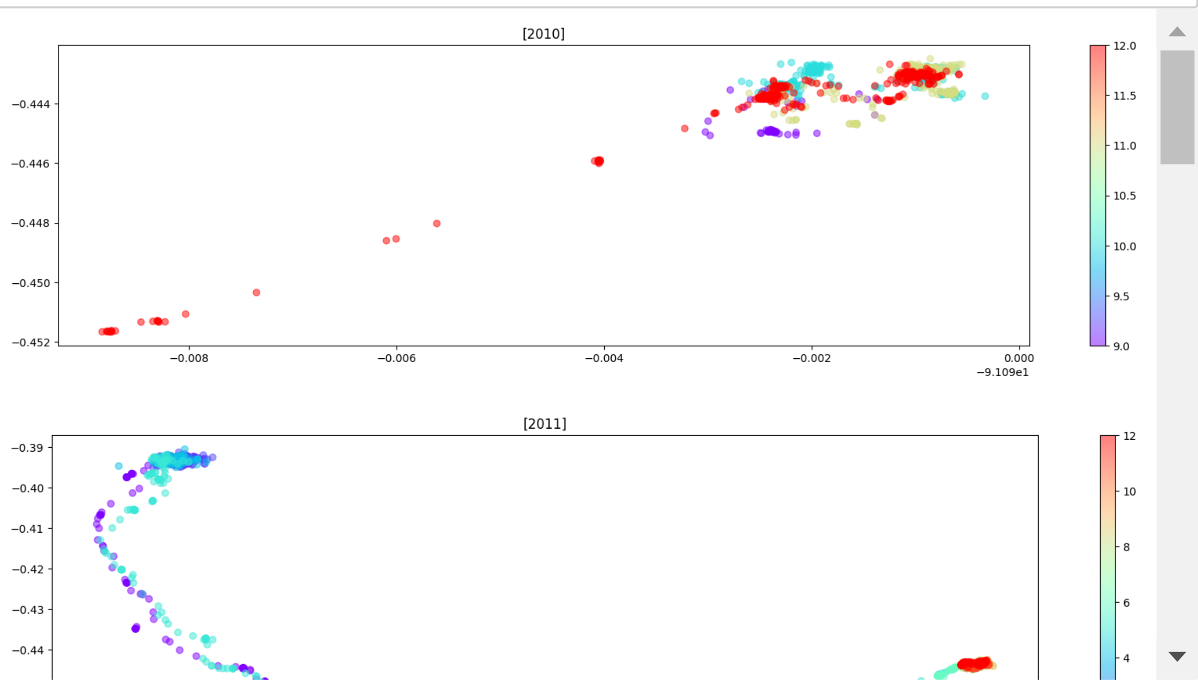
In [44]:
```
plt.figure(figsize = (20,5))

ax = plt.axes()
plt.scatter(christian.location_long.values, christian.location_lat.values,c=ch
plt.title('Generalized Christian Migration')
plt.colorbar()
```

Out[44]: <matplotlib.colorbar.Colorbar at 0x1a9b3cffbb0>



Again, same process of splitting up the years for Christian just like for Alison.

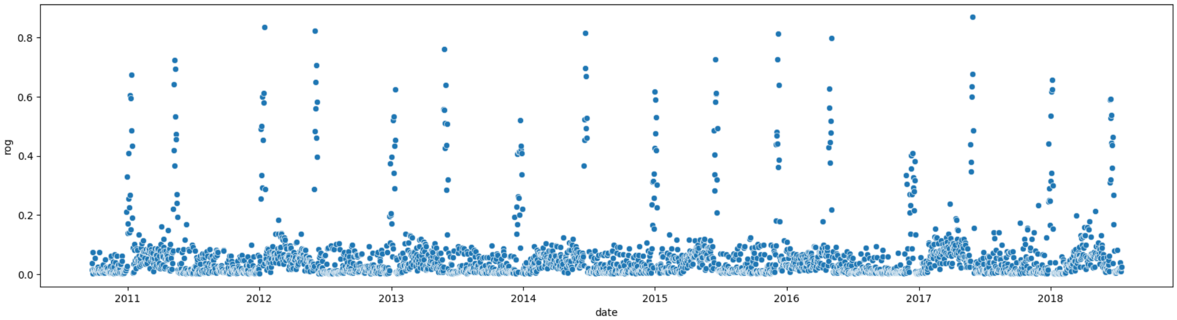
In [45]:
```
for x in christian_n_years:
    plt.figure(figsize = (20,5))
    ax = plt.axes()
    plt.scatter(x.location_long.values, x.location_lat.values,c=x['date_month'
    plt.title(x.date_year.unique())
    plt.colorbar()
```



- 2010 is again not a great year to look at when trying to track migration patterns.

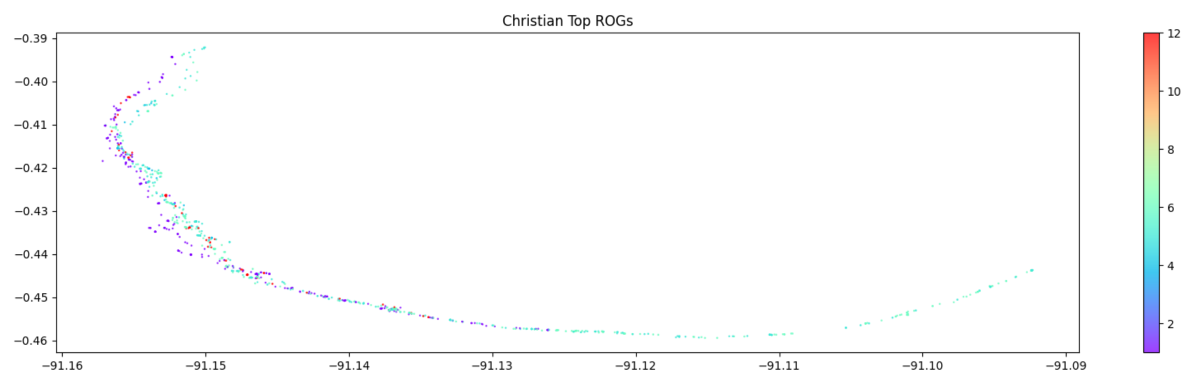
In [46]:
```
plt.figure(figsize = (20,5))
sns.scatterplot(data = christian_radius_gyration, x="date", y="rog")
```

Out[46]: <Axes: xlabel='date', ylabel='rog'>



In [47]:
```
#Point size corresponds to the size of the rogs
plt.figure(figsize = (20,5))
ax = plt.axes()
plt.scatter(christian_top_rogs.location_long.values, christian_top_rogs.locati
plt.title('Christian Top ROGs')
plt.colorbar()
```

Out[47]: <matplotlib.colorbar.Colorbar at 0x1a9b26146d0>



While it is a little less evident, it can still be seen that Christian's migration endpoints are the top left and the bottom right of the graph. Again, this becomes evident when comparing Christian's generalized migration map to his top rog map.

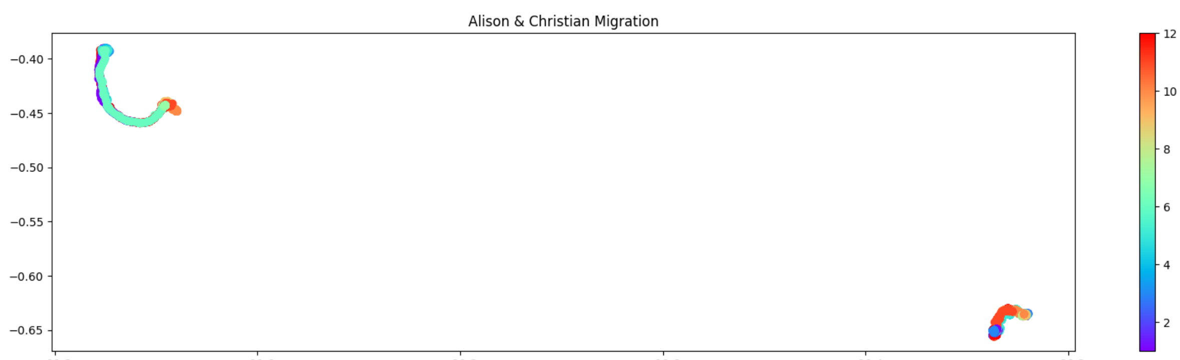
In [48]:
```
map_f = christian_trajectory.query("year == 2012").plot_trajectory(zoom=14, st
christian_trajectory.query("year == 2013").plot_trajectory(map_f=map_f, start_
christian_trajectory.query("year == 2014").plot_trajectory(map_f=map_f, start_
christian_trajectory.query("year == 2015").plot_trajectory(map_f=map_f, start_
christian_trajectory.query("year == 2016").plot_trajectory(map_f=map_f, start_
map_f
```

Out[48]: Make this Notebook Trusted to load map: File -> Trust Notebook

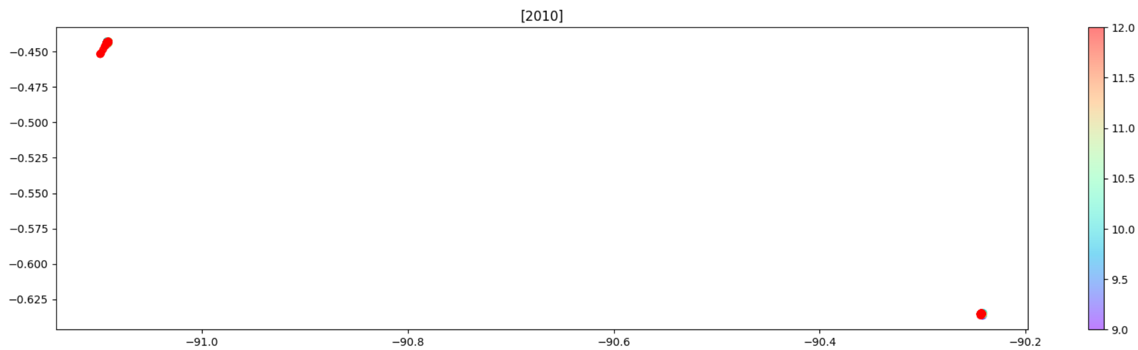
Alison and Christian

In [49]:
```
plt.figure(figsize = (20,5))
ax = plt.axes()
plt.scatter(al_ch.location_long.values, al_ch.location_lat.values,c=al_ch['dat
plt.title('Alison & Christian Migration')
plt.colorbar()
```

Out[49]: <matplotlib.colorbar.Colorbar at 0x1a9c39fca90>

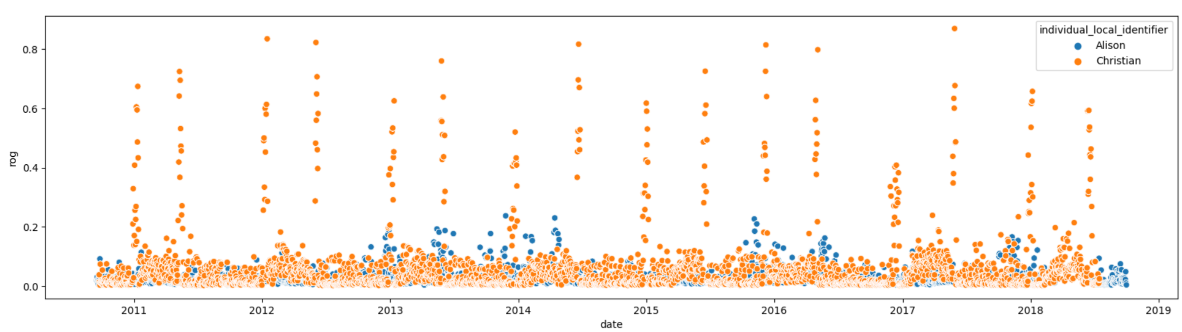


In [50]:
```
for x in al_ch_n_years:
    plt.figure(figsize = (20,5))
    ax = plt.axes()
    plt.scatter(x.location_long.values, x.location_lat.values,c=x['date_month'
    plt.title(x.date_year.unique())
    plt.colorbar()
```



In [51]:
```
plt.figure(figsize = (20,5))
sns.scatterplot(data = al_ch_n, x="date", y="rog",hue="individual_local_identi
```

Out[51]: <Axes: xlabel='date', ylabel='rog'>



In [52]:
```
map_f = alison_trajectory.query("year == 2012").plot_trajectory(zoom=14, start
alison_trajectory.query("year == 2013").plot_trajectory(map_f=map_f, start_enc
alison_trajectory.query("year == 2014").plot_trajectory(map_f=map_f, start_enc
alison_trajectory.query("year == 2015").plot_trajectory(map_f=map_f, start_enc
alison_trajectory.query("year == 2016").plot_trajectory(map_f=map_f, start_enc
christian_trajectory.query("year == 2013").plot_trajectory(map_f=map_f, start_
christian_trajectory.query("year == 2014").plot_trajectory(map_f=map_f, start_
christian_trajectory.query("year == 2015").plot_trajectory(map_f=map_f, start_
christian_trajectory.query("year == 2016").plot_trajectory(map_f=map_f, start_
map_f
```

Out[52]: Make this Notebook Trusted to load map: File -> Trust Notebook

Story and Conclusion

Conclusions for Alison

- As mentioned above, the two most likely endpoints for Alison's migrations are the following:
  1. Between -90.275 and -90.270 (ME1)
  2. Between -90.245 and -90.240 (ME2)
- We can be fairly confident that those are truly the migration end points because, based on Alison's radii of gyration, she moves the least in those areas. If she were to be more active in both of those areas, it may call into question whether those are actually endpoints for her.
- Looking at Alison's geographical map, we can see that she moves in between an area just north of El Cascajo (we'll call this ME1 for migration endpoint 1), and northeast of the city (ME2).
- From the breakdown of Alison's migration patterns by year, we gain very little from years 2010 and 2011. However, we can see a few patterns from Alison's migration trajectories in

the other years.
   1. Alison is capable of migrating to and from an area more than once in a year. This is evident in 2013 because Alison seemed to have spent both the beginning and the end of the year in ME1 and she stayed in ME2 during the middle of the year. However, by looking at the other graphs, this is not typical.
   2. Typically, if Alison starts in ME1, she will begin and end her migration in the first half of the year, (March through May).
   3. If Alison begins the year in ME2, she will start migrating in the second half of the year, closer to August or September and end around November or December.
- From her migration patterns, it seems that ME2 is a more attractive place for Alison to stay.

Conclusion for Christian

- Repeating some of the process for Christian, it can be seen that his most likely migration endpoints (we will call them ME3 and ME4) are the following (all numbers represent the longitude coordinate):
   1. ME3 is the cluster around -91.15
   2. ME4 is the cluster between -91.0975 and -91.09
- It seems like Christian always begins his year in ME3 and then migrates to ME4 within the first half of the year (typically before May). Then, he stays in ME4 until November or December, when he migrates back to ME3 to begin the new year. This seems to be the most likely case, but I did make an assumption that he travels back to ME3 at the end of the year, because it is not incredibly apparent. For some of the yearly graphs, Christian seems to spend the end of the year in ME4, and then begin the new year in ME3. However, this would imply that Christian has the capability of teleportation, which I doubt. Some of the later yearly graphs do show some migration back from ME4 to ME3 at the end of the year, so I decided (somewhat arbitrarily) that it was safer to assume that Christian migrates back to ME3 at the end of each year.

Comparisons Between the Two

- The largest difference between Alison's and Christian's migration patterns are that Alison (typically) makes only one trip per year. She starts in either ME1 or ME2 and ends in the opposite one. She will repeat this process the next year, just starting from where she ended. Christian, however, will make two trips in one year. He will always end where he starts (that starting point being ME3).
- Another difference is that Christian has consistent migration months whereas Alison changes her migration months depending on her starting point. This does, however, lead to an interesting similarity between the two in that they both stay longer in the east migration endpoint than the west. I am unsure why this is the case, but it should be noted.
- The fact that Christian is more mobile than Alison is also supported by the Alison's and Christian's combined date vs rog graph. All of Christians highest rogs are significantly higher than Alisons, indicating that he moves more often than Alison because he has more distance to travel.

Critiques and Points to Improve On

There are several points that could be improved upon for this project.

1. More time could have been spent looking at (or color coding) the "heading" for each event. This may have been valuable to look at because, hypothetically, if there are areas where the tortoises seem to be moving back and forth (based on their heading), then that could be an indicator of a resting point. Additionally, if there is a strong amount of single direction heading, then that would likely correspond with migration.
2. Speed could have been factored in. If a tortoise had consistent headings but they had very slow speed relative to their other speeds, that could narrow down what I consider to be a migration versus what is not.
3. I Could have created some histograms to determine whether there was a favorite movement time for Alison and Christian, but this would have taken more understanding of the data related to when they are actually migrating (i.e. the first two points).
4. Finally, and possibly the most important critique for this project, I could have looked at two turtles who were migrating in a similar area. This may have allowed me to make stronger visual comparisons between the two. It is possible that Alison is an oddball in her migration patterns, but she was not migrating in the same area or group as Christian, so there were limited points of comparison between the two. The distance between Alison and Christian was also large enough that it made my graphical color-coding less valuable. The points for Christian and Alison were so close together (respectively) as a result of the tortoises' distance between one another, that it was hard to tell what the actual migration patterns were between the two. There was likely a more effective way to compare the two on the same graph. I have doubts about the effectiveness of my multi-tortoise graphs.

In [ ]:

```

```