In [1]:
```
!pip install global_land_mask --user
```

```
DEPRECATION: Python 3.4 support has been deprecated. pip 19.1 will be the last one supporting it. Please upgrade your
Python as Python 3.4 won't be maintained after March 2019 (cf PEP 429).
Requirement already satisfied: global_land_mask in /home/codio/.local/lib/python3.4/site-packages (1.0.0)
```

In [2]:
```
import ipywidgets as widgets
from ipywidgets import interact, interact_manual
```

In [3]:
```
# Importing libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from plotly import tools
# import plotly.plotly as py
import chart_studio.plotly
from plotly.offline import init_notebook_mode, iplot
init_notebook_mode(connected=True)
import plotly.graph_objs as go
import plotly.figure_factory as ff
from IPython.display import HTML, Image
import plotly.express as px
import numpy
from global_land_mask import globe
from math import *
px.set_mapbox_access_token(open(".mapbox_token").read())
```

In [4]:
```
birds = pd.read_csv("bird_tracking.csv")
```

Are there differences in the altitudes of the Birds? Are there futher differences when factoring in time of day?

First, we establish "night" as 6pm to 5am, with day being everything else -- we make a column in the data frame to specify the time of day as "night" or "day", and additionally make separate data frames for day and night values.

Next, we create a new column for altitude, compressing our altitude values down to a more intuitive scale by utilizing the log function. All altitudes equal to or less than zero are simply put as 1.

In [5]:
```
birds["Time of Day"] = birds.apply(lambda birds: "Night" if ((pd.to_datetime(birds["date_time"]).hour >= 18) | (pd.to_da
birds_night = birds[birds["Time of Day"] == "Night"]
birds_day = birds[birds["Time of Day"] == "Day"]

def convert_log(value):
    return numpy.log2(value)

birds["altitude size marker"] = birds.apply(lambda d: convert_log(d["altitude"]) if d["altitude"] > 0 else 1, axis = 1)
```

For analysis, we determine the average altitude for each bird in general, during the day, and during the night.

In [6]:
```
birds.groupby("bird_name")["altitude"].mean().to_frame().reset_index()
```

Out[6]:

	bird_name	altitude
0	Eric	60.249406
1	Nico	67.900478
2	Sanne	29.159922

In [7]:
```
birds_day.groupby("bird_name")["altitude"].mean().to_frame().reset_index()
```

Out[7]:

	bird_name	altitude
0	Eric	67.478760
1	Nico	77.119432
2	Sanne	30.493537

In [8]:
```
birds_night.groupby("bird_name")["altitude"].mean().to_frame().reset_index()
```

Out[8]:

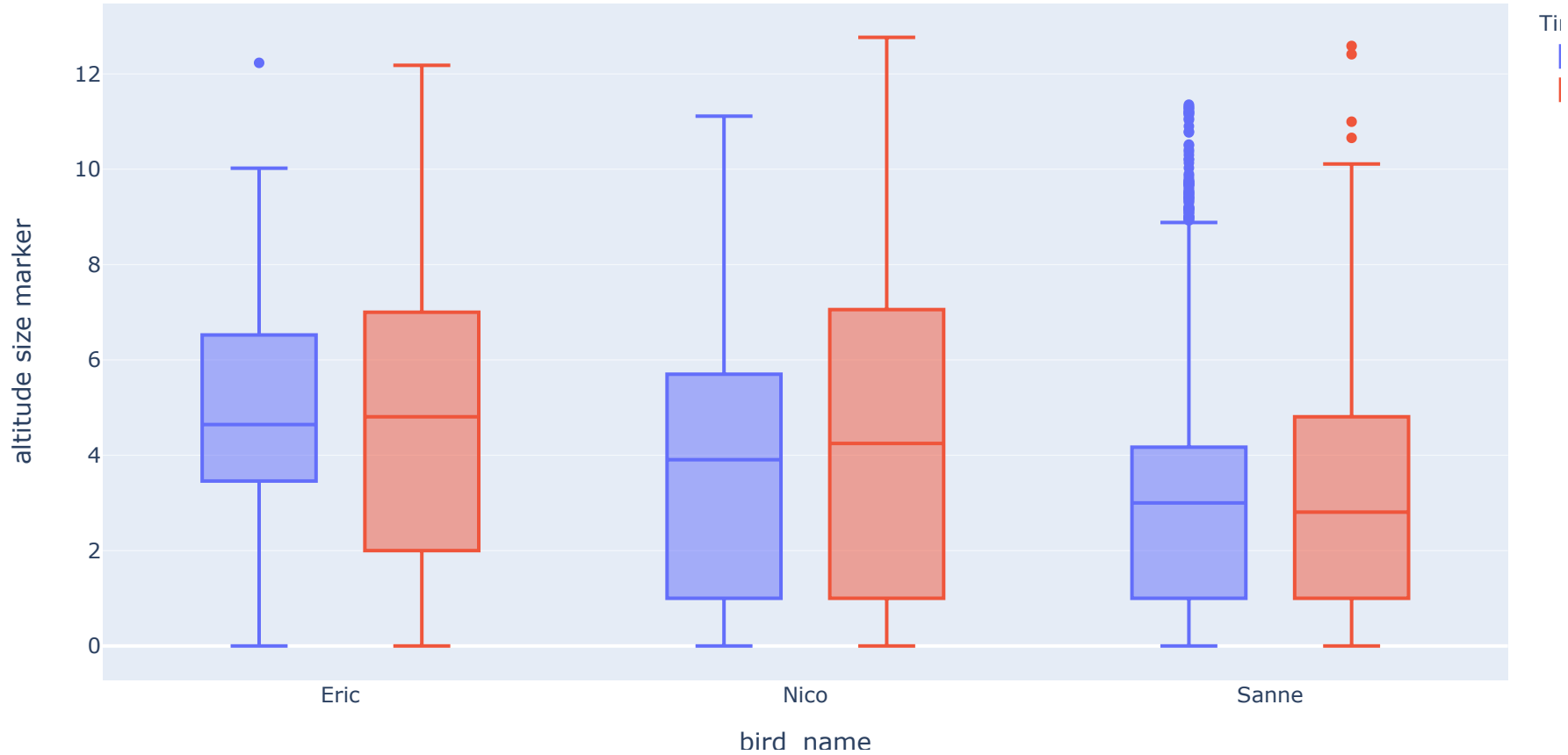
	bird_name	altitude
0	Eric	50.772589
1	Nico	55.696446
2	Sanne	27.385665

Clearly, Nico likes to fly the highest, with Eric not too far behind. Sanne, compared to the other birds, likes to fly at much lower altitudes.

When factoring in day or night, it is clear that Nico and Eric fly lower during the night. Sanne, on the other hand, appears to not have a preference for altitude depending on time of day.

We can illustrate these themes in a boxplot

In [10]:
```
fig_altitude_time = px.box(birds, x = "bird_name", y = "altitude size marker", color = "Time of Day")
fig_altitude_time.show()
```



Now, let's do the same thing for speed.

In [11]:
```
birds.groupby("bird_name")["speed_2d"].mean().to_frame().reset_index()
```

Out[11]:

	bird_name	speed_2d
0	Eric	2.300545
1	Nico	2.908726
2	Sanne	2.450434

In [12]:
```
birds_day.groupby("bird_name")["speed_2d"].mean().to_frame().reset_index()
```

Out[12]:

	bird_name	speed_2d
0	Eric	2.589002
1	Nico	3.175397
2	Sanne	2.593278

In [13]:
```
birds_night.groupby("bird_name")["speed_2d"].mean().to_frame().reset_index()
```

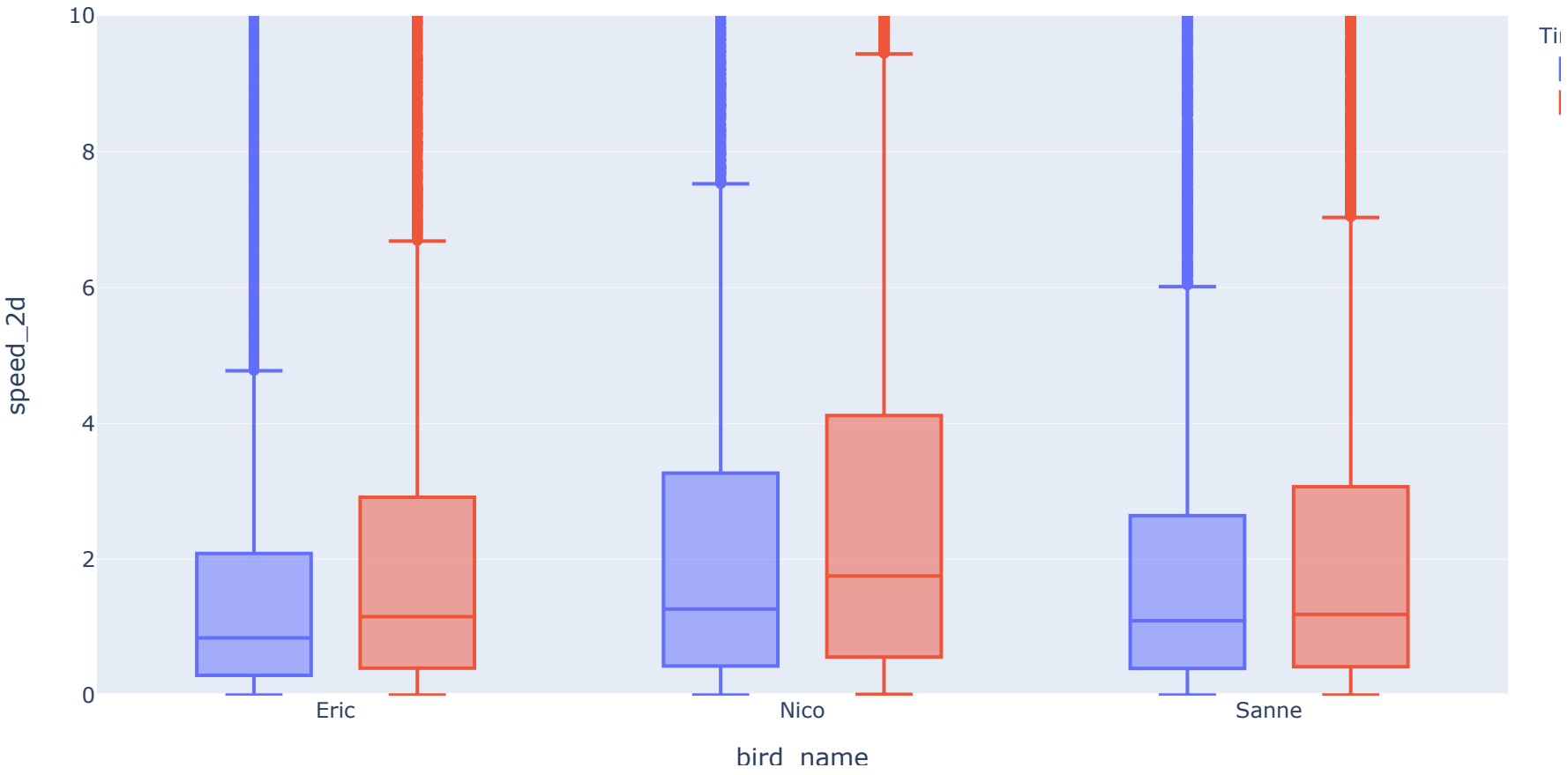
Out[13]:

	bird_name	speed_2d
0	Eric	1.922081
1	Nico	2.555752
2	Sanne	2.259649

Nico appears to travel the fastest out of any of the birds, with Sanne next and Eric being the slowest. All the birds, travel faster during the day than during the night, but again, Sanne appears to have less of a preference depending on the time of day than the other two birds.

Again, we can visualize this in a boxplot. Since there are many outliers for speed, we limit the y-axis to just range 1 to 10.

In [14]:
```
fig_speed_time = px.box(birds, x = "bird_name", y = "speed_2d", color = "Time of Day")
fig_speed_time.update_layout(yaxis_range=[0,10])
fig_speed_time.show()
```



What about the maximums?

In [15]:
```
birds.groupby("bird_name")["speed_2d"].max().to_frame().reset_index()
```

Out[15]:

	bird_name	speed_2d
0	Eric	63.488066
1	Nico	48.381510
2	Sanne	57.201748

In [16]:
```
birds_day.groupby("bird_name")["altitude"].max().to_frame().reset_index()
```

Out[16]:

	bird_name	altitude
0	Eric	4639
1	Nico	6965
2	Sanne	6145

Interestingly, there seems to be an inverse relationship with the maximums of the birds' speed and altitude. Eric has the fastest recorded speed, but the lowest maximum altitude. Conversely, Nico has the highest recorded altitude, but the lowest maximum speed. Lastly, Sanne places in the middle with regards to both.

Furthermore, despite being the slowest bird on average, Eric has the fastest maximum speed. Also, despite being the lowest flying bird on average, Sanne does not have the lowest maximum altitude.

Does Terrain Type affect speed and altitude for each bird?

Here, we add a new column to the data frame based on the function below, which takes input coordinates and returns if the location is land. Thus, we can make a new column telling us whether each point is over land or over water, and make separate data frames based off this condition as well.

In [17]:
```
def over_water(lat, long):
    if globe.is_land(lat, long) == True:
        return "Over Land"
    else:
        return "Over Water"

birds["Terrain"] = birds.apply(lambda x: over_water(x["latitude"], x["longitude"]), axis = 1)
birds_land = birds[birds["Terrain"] == "Over Land"]
birds_water = birds[birds["Terrain"] == "Over Water"]
```

In [18]:
```
birds_land.groupby("bird_name")["speed_2d"].mean().to_frame().reset_index()
```

Out[18]:

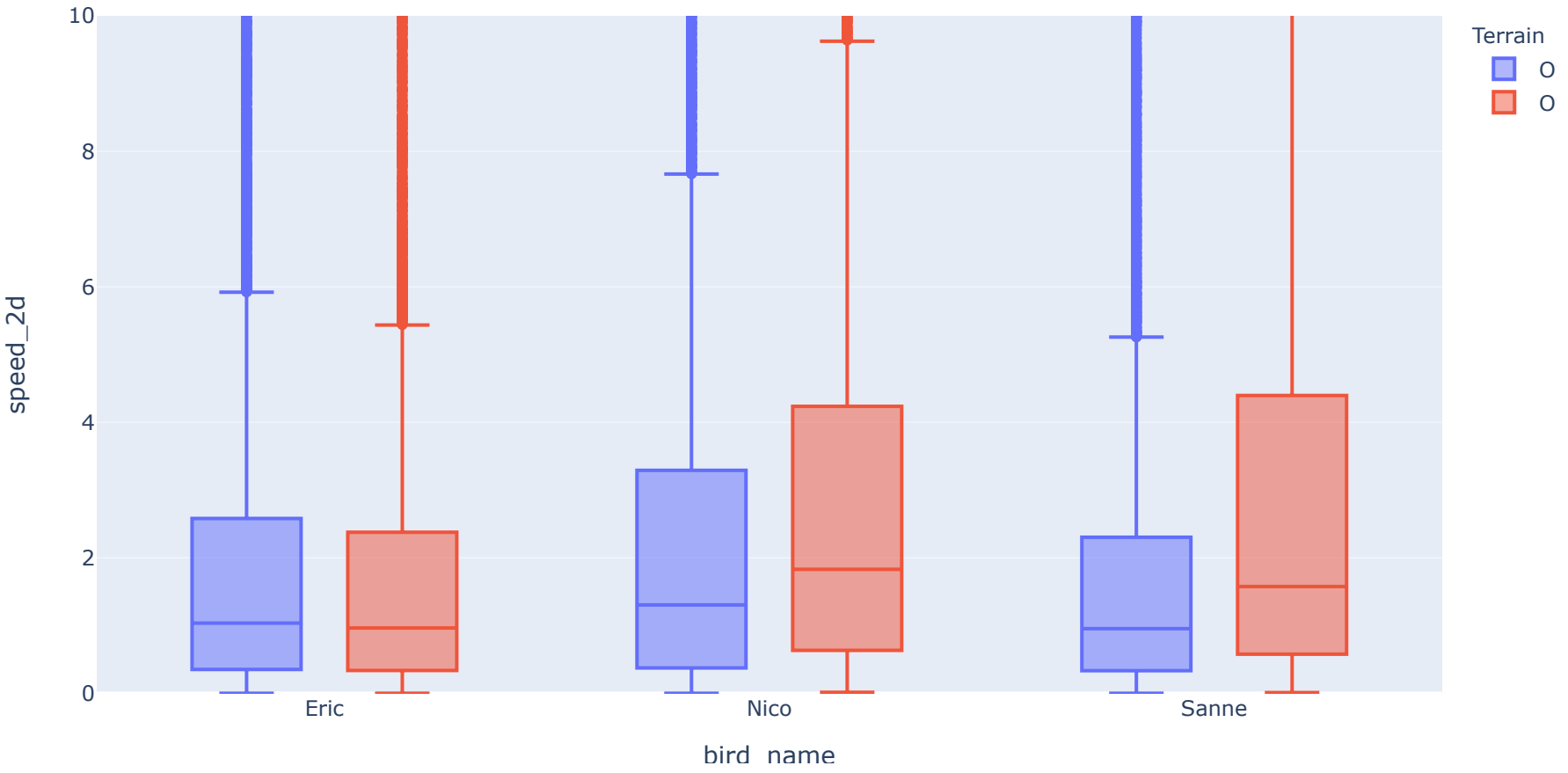
	bird_name	speed_2d
0	Eric	2.416903
1	Nico	2.646650
2	Sanne	2.076279

In [19]:
```
birds_water.groupby("bird_name")["speed_2d"].mean().to_frame().reset_index()
```

Out[19]:

	bird_name	speed_2d
0	Eric	2.160883
1	Nico	3.195355
2	Sanne	3.117444

In [20]:
```
fig_speed_terrain = px.box(birds, x = "bird_name", y = "speed_2d", color = "Terrain")
fig_speed_terrain.update_layout(yaxis_range=[0,10])
fig_speed_terrain.show()
```



Over land, Sanne is by far the slowest average bird, which is somewhat surprising because Sanne was the middle bird with overall speed. Eric is the only bird that travels faster over land than over water, as the other birds travel much faster over water. Nico places as the fastest bird in each category, which makes sense considering Nico is the fastest bird overall on average.

Now let's see altitude.

In [21]:
```
birds_land.groupby("bird_name")["altitude"].mean().to_frame().reset_index()
```

Out[21]:

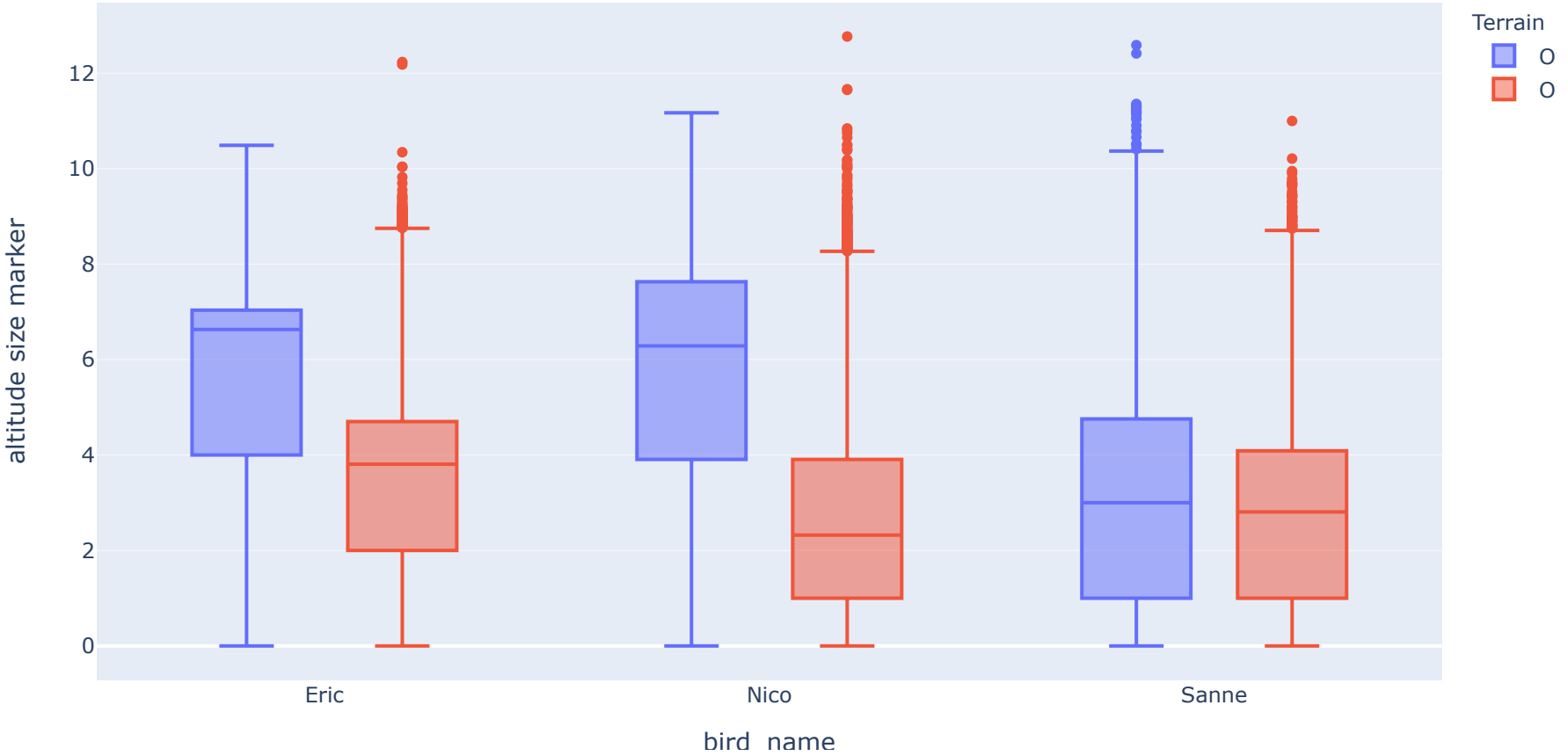
	bird_name	altitude
0	Eric	88.355298
1	Nico	112.883111
2	Sanne	34.335213

In [22]:
```
birds_water.groupby("bird_name")["altitude"].mean().to_frame().reset_index()
```

Out[22]:

	bird_name	altitude
0	Eric	26.469188
1	Nico	18.834587
2	Sanne	19.891709

In [26]:
```
fig_altitude_terrain = px.box(birds, x = "bird_name", y = "altitude size marker", color = "Terrain")
fig_altitude_terrain.show()
```



Nico, being the bird that flys highest on average, unsurprisingly flys the highest on average while over land. However, Nico flys over water at the lowest altitude on average. Again, we can see that out of all the birds, Sanne has the least variation of altitude. There is not a huge difference between Sanne's average altitudes over land or over water. But, it is clear that all the birds do fly lower over water to some extent.

So, to recap:

Nico:

-On average, travels at the highest altitude and highest speed of all 3 birds -Has the highest maximum altitude, but surprisingly has the lowest maximum speed -Highest average altitude over land, but the lowest average altitude over water -Has the highest average speed over land and over water

Eric:

-Places in the middle in average altitude, both during the day and night -Slowest bird on average (regardless of time of day), yet has the fastest maximum speed -Lowest maximum altitude by a wide margin, despite average altitude placing in the middle -Only bird to travel faster over land than over water -Places in the middle for average altitude over land, but has the highest average altitude over water

Sanne:

-Lowest average altitude regardless of time of day by a huge margin, yet has 2nd highest maximum altitude -Places in the middle for speed regardless of time of day, and places in the middle for maxiumum speed as well -By far the slowest bird on average over land, but essentially tied for the fastest bird on average over water -By far the lowest flying bird on average over land, but is not the lowest flying bird on average over water

Factoring in Distances

To examine any possible differences in the distances covered by the birds, we can make a data frame comparing distances and time of day.

To do this, we can build functions to calculate distances between points, and furthermore calculate the absolute value of the distance covered by each bird.

In [23]:
```
def distance(lon1, lat1, lon2, lat2):
    lon1, lat1, lon2, lat2 = map(radians, [lon1, lat1, lon2, lat2])
    dlon = lon2 - lon1
    dlat = lat2 - lat1
    a = sin(dlat/2)**2 + cos(lat1) * cos(lat2) * sin(dlon/2)**2
    c = 2 * asin(sqrt(a))
    r = 6371
    return c * r

def get_dist_bird(bird, df):
    new_df = df[df["bird_name"] == bird]
    new_df = new_df.assign(date=lambda d: pd.to_datetime(d['date_time']))
    new_df["change_dist"] = 0
    new_df = new_df.reset_index()
    for row in range(len(new_df)-1):
        change_dist = distance(new_df.loc[row, "longitude"], new_df.loc[row, "latitude"], new_df.loc[row+1, "longitude"]
        new_df.loc[row+1, "change_dist"] = change_dist
    return round(new_df["change_dist"].sum(),2)
```

In [24]:
```
names = {'Bird Name': ["Eric", "Nico", "Sanne"]}
bird_distance_time = pd.DataFrame(data=names)

bird_distance_time["Distance at Day"] = 0
bird_distance_time["Distance at Day"].iloc[0] = get_dist_bird("Eric", birds_day)
bird_distance_time["Distance at Day"].iloc[1] = get_dist_bird("Nico", birds_day)
bird_distance_time["Distance at Day"].iloc[2] = get_dist_bird("Sanne", birds_day)

bird_distance_time["Distance at Night"] = 0
bird_distance_time["Distance at Night"].iloc[0] = get_dist_bird("Eric", birds_night)
bird_distance_time["Distance at Night"].iloc[1] = get_dist_bird("Nico", birds_night)
bird_distance_time["Distance at Night"].iloc[2] = get_dist_bird("Sanne", birds_night)

bird_distance_time["% Distance at Night"] = 0
bird_distance_time["% Distance at Night"].iloc[0] = round(bird_distance_time["Distance at Night"].iloc[0] / (bird_distar
bird_distance_time["% Distance at Night"].iloc[1] = round(bird_distance_time["Distance at Night"].iloc[1] / (bird_distar
bird_distance_time["% Distance at Night"].iloc[2] = round(bird_distance_time["Distance at Night"].iloc[2] / (bird_distar

bird_distance_time["% Distance at Day"] = 0
bird_distance_time["% Distance at Day"].iloc[0] = round(bird_distance_time["Distance at Day"].iloc[0] / (bird_distance_t
bird_distance_time["% Distance at Day"].iloc[1] = round(bird_distance_time["Distance at Day"].iloc[1] / (bird_distance_t
bird_distance_time["% Distance at Day"].iloc[2] = round(bird_distance_time["Distance at Day"].iloc[2] / (bird_distance_t

bird_distance_time
```

Out[24]:

	Bird Name	Distance at Day	Distance at Night	% Distance at Night	% Distance at Day
0	Eric	14676.25	9884.07	0.40	0.60
1	Nico	22856.21	19222.30	0.46	0.54
2	Sanne	20052.53	19764.10	0.50	0.50

In [32]:
```
name_list = ['Eric','Nico','Sanne','all']
```

In [38]:
```
@interact(name = name_list)
def map1(name):
    if name == 'all':
        df1 = birds
    else:
        df1 = birds_day[birds_day['bird_name'] == name]
    map_distance_day = px.scatter_mapbox(df1, lat="latitude", lon="longitude", color = "bird_name",
                        color_continuous_scale=px.colors.cyclical.IceFire, size_max= 7, zoom=2)
    map_distance_day.show()
```

name Eric

In [36]:
```
@interact(name = name_list)
def map2(name):
    if name == 'all':
        df1 = birds
    else:
        df1 = birds_night[birds_night['bird_name'] == name]
    map_distance_night = px.scatter_mapbox(df1, lat="latitude", lon="longitude", color = "bird_name",
                        color_continuous_scale=px.colors.cyclical.IceFire, size_max= 7, zoom=2)
    map_distance_night.show()
```

name Eric

As shown, Sanne is the only bird that does not have a higher percentage of distance covered during the day than during the night. Also, Eric has the least amount of distance covered at day and at night by far

Now let's do the same process but for distance by terrain type.

In [25]:
```
names = {'Bird Name': ["Eric", "Nico", "Sanne"]}
bird_distance_terrain = pd.DataFrame(data=names)

bird_distance_terrain["Distance Over Land"] = 0
bird_distance_terrain["Distance Over Land"].iloc[0] = get_dist_bird("Eric", birds_land)
bird_distance_terrain["Distance Over Land"].iloc[1] = get_dist_bird("Nico", birds_land)
bird_distance_terrain["Distance Over Land"].iloc[2] = get_dist_bird("Sanne", birds_land)

bird_distance_terrain["Distance Over Water"] = 0
bird_distance_terrain["Distance Over Water"].iloc[0] = get_dist_bird("Eric", birds_water)
bird_distance_terrain["Distance Over Water"].iloc[1] = get_dist_bird("Nico", birds_water)
bird_distance_terrain["Distance Over Water"].iloc[2] = get_dist_bird("Sanne", birds_water)

bird_distance_terrain["% Distance Over Land"] = 0
bird_distance_terrain["% Distance Over Land"].iloc[0] = round(bird_distance_terrain["Distance Over Land"].iloc[0] / (bir
bird_distance_terrain["% Distance Over Land"].iloc[1] = round(bird_distance_terrain["Distance Over Land"].iloc[1] / (bir
bird_distance_terrain["% Distance Over Land"].iloc[2] = round(bird_distance_terrain["Distance Over Land"].iloc[2] / (bir

bird_distance_terrain["% Distance Over Water"] = 0
bird_distance_terrain["% Distance Over Water"].iloc[0] = round(bird_distance_terrain["Distance Over Water"].iloc[0] / (b
bird_distance_terrain["% Distance Over Water"].iloc[1] = round(bird_distance_terrain["Distance Over Water"].iloc[1] / (b
bird_distance_terrain["% Distance Over Water"].iloc[2] = round(bird_distance_terrain["Distance Over Water"].iloc[2] / (b

bird_distance_terrain
```

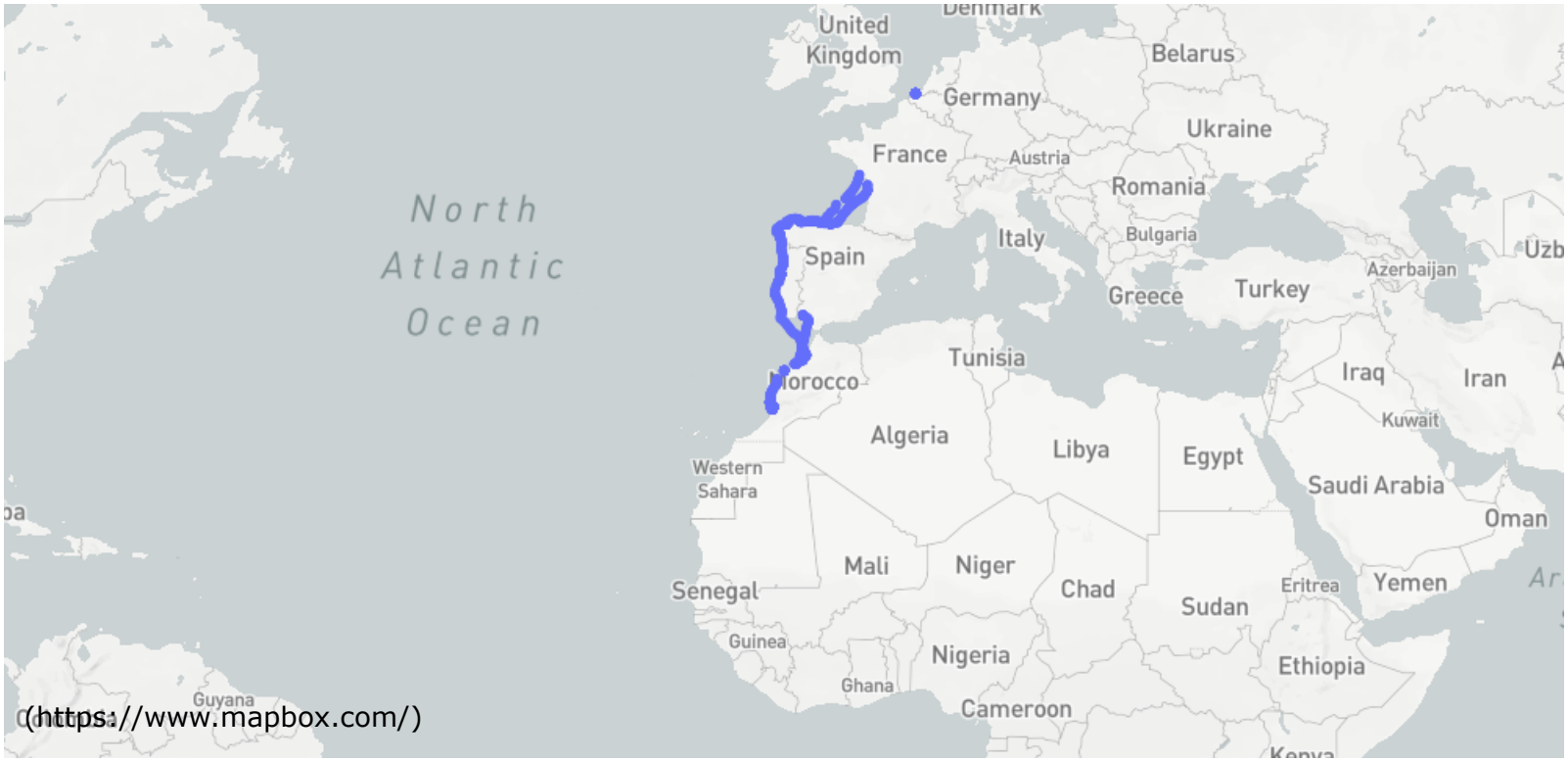
Out[25]:

	Bird Name	Distance Over Land	Distance Over Water	% Distance Over Land	% Distance Over Water
0	Eric	14590.36	8240.15	0.64	0.36
1	Nico	17746.81	19763.95	0.47	0.53
2	Sanne	16106.24	19811.27	0.45	0.55

In [29]:
```
@interact(name = name_list)
def map3(name):
    if name == 'all':
        df1 = birds
    else:
        df1 = birds_land[birds_land['bird_name'] == name]
    map_distance_land = px.scatter_mapbox(df1, lat="latitude", lon="longitude", color = "bird_name",
                        color_continuous_scale=px.colors.cyclical.IceFire, size_max= 7, zoom=2)
    map_distance_land.show()
```

In [37]:
```
@interact(name = name_list)
def map4(name):
    if name == 'all':
        df1 = birds
    else:
        df1 = birds_water[birds_water['bird_name'] == name]
    map_distance_water = px.scatter_mapbox(df1, lat="latitude", lon="longitude", color = "bird_name",
                        color_continuous_scale=px.colors.cyclical.IceFire, size_max= 7, zoom=2)
    map_distance_water.show()
```

name Eric



As shown, Eric is the only bird to cover more distance over land than over water. However, he ironically has the lowest total distance covered over land. Nico and Sanne both have similar amounts of distance covered over land and over water, as well as percentages for these conditions.