Spatiotemporal Data:

Tracking Fruit Bats in Ghana

Sarah Jaraha

Welcome!

This workshop introduces spatiotemporal data using PostgreSQL and QGIS. The dataset used in this workshop tracks the flight paths of nine fruit bats in Accra, Ghana. By the end of the workshop, workers will have:

1. imported spatiotemporal data
2. processed spatiotemporal data
3. run queries against spatiotemporal data

What is Spatiotemporal Data?

Spatiotemporal data stores coordinates and time measurements within a single geometry. Without spatiotemporal data, time measurements would need to be stored in a separate table. Storing spatial and temporal data together allows for better organization and more efficient analysis of the data. It can be used to monitor flight paths, track the spread of disease, or to document animal movements.

Components of Spatiotemporal Geometries:

1. *measurement coordinate (M-coordinate):* An M-coordinate can store various types of measurements. For spatiotemporal data, the measurement is always time. In PostgreSQL, the M-coordinate is embedded within the geometry field. The function ST_M() is used to retrieve the M-coordinate from a geometry.
2. *geometry*: The geometry stores location coordinates and M-coordinates. Geometries that store M-coordinates have a suffix "M". pointM and linestringM are examples of geometries that can store M-coordinates.
3. *SRID:* The SRID provides information on the coordinate system, projection, and datum of the coordinates. The raw data for this workshop was collected with GPS and is resultingly stored in WGS84. WGS84 is not suited for measuring distance for two reasons: it uses a global coordinate system, and it returns measurements in decimal degrees (and incomprehensible unit). To measure distance, the SRID must be transformed to a local coordinate system. This is discussed furthur in later sections.

Getting Started

Data Download:

1. Go to www.movebank.org
2. Click "Browse Tracks"
3. In the search bar, type "fruit bats in ghana". A study called "Fruit Bats in Ghana (data from Sapir et al.2014)" should appear at the bottom of the search results.
4. Click the "i" icon next to the appropriate study, then select "open in studies page".
5. Select the download tab, then click "download data".
6. Agree to the terms and conditions and download the data in CSV format.

Import to PostgreSQL:

1. Open the database manager in QGIS. (*this workshop assumes that the appropriate database connections exist between QGIS and PostgreSQL*)
2. With the desired database connection selected, open the "Import vector layer" dialogue box.
3. Select the "convert field names to lowercase" box. Name the table **ghana_fruit_bats**. Select "OK" to import the CSV to PostgreSQL.

Preparing to Process the Data

1. In PostgreSQL, run the following query against the database containing the ghana_fruit_bats table to create the PostGIS extension. The PostGIS extension allows spatial queries to be run:

```
CREATE EXTENSION postgis;
```

2. In the object browser, navigate to the ghana_fruit_bats table and expand the column list. Change the **dashes to underscores** for the following columns: *location-long, location-lat*, and *individual-local-identifier*. (This can be done by right clicking the column name, selecting "Properties", and editing the column name in the "Name" field).

Processing the Data

1. Create a table to store the spatiotemporal geometries by running the following query:

```
CREATE TABLE ghana_fruit_bats_m
(bat_id integer PRIMARY KEY,
line_geom geometry (LINESTRINGM, 4326));
```

   The table has two columns: bat_id and line_geom. bat_id is the primary key. Each row in bat_id will hold the identifier for an individual bat. line_geom will store the spatiotemporal geometries.

   The column definition for line_geom includes the data type (geometry), the geometry type (linestringM), and the local SRID (4326). The geometry type must be specified to allow storage of M-coordinates. Each row in line_geom will represent the flight path for an individual bat.

2. Insert data from ghana_fruit_bats into ghana_fruit_bats_m by running the following query:

```
INSERT INTO ghana_fruit_bats_m
(SELECT
individual_local_identifier::integer as bat_id, ST_MakeLine(sub.pointm) as
line_geom
    FROM
    (SELECT individual_local_identifier,
        ST_SetSRID(
            ST_MakePointM(location_long::double precision, -----make
pointM
                location_lat::double precision,
                EXTRACT(epoch FROM timestamp::timestamp)),4326) as pointm
    FROM ghana_fruit_bats
    ORDER BY EXTRACT(epoch FROM timestamp::timestamp)) as sub  ----order
by timestamp
GROUP BY sub.individual_local_identifier   -------separate by bat
ORDER BY sub.individual_local_identifier);  ---------order by bat
```

   Let's take a closer look at this query to understand what is happening...

A Closer Look

In this section, the query from step two, above, will be dissected and explained. We will start from the innermost piece, then work outward.

[I] *Formatting the timestamp*

```
EXTRACT(epoch FROM timestamp::timestamp)),4326) as pointm
```

Initially, the timestamp column is stored as a varchar data type. To make use of the timestamp column, it must be transformed to the epoch data type. This is done by casting the timestamp column to a timestamp data type, then using an EXTRACT() function to convert from timestamp to epoch.

Epoch represents time as the number of seconds that have passed since January 1, 1970. This is also called Continental Universal Time, or unix time. Storing time measurements as epoch allows for ordering and use as an M-coordinate.

[II] *Creating pointM geometries*

```
ST_SetSRID(
    ST_MakePointM(
        location_long::double precision,
        location_lat::double precision,
        EXTRACT(epoch FROM timestamp::timestamp)),4326)
as pointm
```

The ST_MakePointM function outputs a pointM geometry type when given longitude, latitude, and M-coordinate. ST_SetSRID was used to assign the SRID of the raw data (WGS84 SRID: 4326). To measure distances, the SRID must be transformed to the local SRID (UTM 30N SRID: 32630). This is done using the ST_Transform function in the "Querying Spatiotemporal Data" section of the workshop.

[III] *Ordering by epoch*

```
ORDER BY EXTRACT(epoch FROM timestamp::timestamp)
```

Before the pointM geometries can be used to make linestringM geometries, they must be ordered by epoch. This ensures that the pointM geometries are connected chronologically.

[IV] *Creating linestringM geometries*

```
ST_MakeLine(sub.pointm) as line_geom
```

After the pointM geometries are ordered, they can be used with ST_MakeLine(). ST_MakeLine() outputs a linestringM when given a series of pointMs.

[V] *Grouping by individual_local_identifier*

```
GROUP BY sub.individual_local_identifier
```

Grouping by the individual_local_identifier separates the linestrings by bat. This way, each linestring will represent the movement of one bat.

Now we're ready to run some queries!

Querying Spatiotemporal Data

Functions

Function	Description
ST_IsValidTrajectory(linestringM)	Returns boolean value indicating whether the M-coordinate increases from one point to the next in a linestringM.
ST_NPoints(geometry)	Returns the number of points in a geometry
ST_Transform(geometry, SRID)	Returns a different geometry with its coordinates transformed to the specified SRID
ST_Length(linestring)	Returns the length
ST_PointN(linestring, integer)	Returns the nth point of a linestring
ST_M(pointM)	Returns the M-coordinate of a point
to_timestamp(epoch)	converts epoch to timestamp

Q: Are the trajectories valid?

```
SELECT bat_id, ST_IsValidTrajectory(line_geom)
FROM ghana_fruit_bats_m;
```

A result of "t" means that the M-coordinate value increases along the linestring. In other words, a result of "t" varifies that the points in the linestring have been connected in chronological order. A result of "f" means something is out of order.

Q: How many observations were made for each bat?

```
SELECT bat_id, ST_NPoints(line_geom)
FROM ghana_fruit_bats_m;
```

ST_NPoints() returns the number of points in a linestring. This can also be interpreted as the number of observations made for each bat. The query shows that bat_id 1607 had the most observations (211).

Q: How far did the each bat fly?

```
SELECT bat_id,
(ST_Length(ST_Transform(line_geom, 32630))/1000)::INTEGER as
flight_distance_km
FROM ghana_fruit_bats_m;
```

A distance measurement! This is where the SRID really comes into play. The data is stored in WGS84 SRID: 4326. However, performing distance calculations in WGS84 will result in incomprehensible results because WGS84 measures distance in decimal degrees (meaningless as a unit of distance). In the query above, ST_Transform is used to transform the SRID to UTM 30N SRID: 32630 (the SRID for Ghana). Keep in mind that the original dataset (ghana_fruit_bats) still uses WGS84. ST_Transform creates a new geometry with the specified SRID and performs calculations on the new geometry.

Then, ST_Length is used to calculate the length of the linestring, using the appropriate SRID. UTM 30N uses meters as the unit of measurement. So, the results are divided by 1000 to convert to kilometers. Finally, casting to an integer data type removes decimals and makes results easier to read.

Q: When was the 100th point recorded on the path of bat_id 1607?

```
SELECT to_timestamp(ST_M(ST_PointN(line_geom, 100)))
FROM ghana_fruit_bats_m
WHERE bat_id = 1607;
```

Two functions are highlighted in the above query: ST_M(), and ST_PointN(). First, ST_PointN selects the 100th point in the linestring. Then, ST_M() extracts the M-coordinate from the 100th point. Recall that the M-coordinate is in the epoch data type which returns seconds since January 1, 1970. To make sense of the result, the to_timestamp() function is used to convert epoch to timestamp. The where clause specifies the desired bat_id. We find that the 100th point on the path of bat_id 1607 was recorded on February 2, 2011 at 2:50 AM GMT.

Further Exploration

Advanced Queries

Function	Description
ST_DistanceCPA(linestring, linestring)	Returns the closest point of approach (CPA) between two moving objects
ST_CPAWithin(linestring, linestring, integer)	Returns a boolean value that indicates whether the two moving objects fall within a specified distance of each other

Further Reading

ST_DistanceCPA: https://postgis.net/docs/ST_DistanceCPA.html

ST_CPAWithin: https://postgis.net/docs/ST_CPAWithin.html

date and time data types: http://www.postgresqltutorial.com/postgresql-timestamp/

related tutorial: http://themagiscian.com/2016/08/05/hooded-vulture-study-using-movebank-kml/

SRID: http://desktop.arcgis.com/en/arcmap/10.3/manage-data/using-sql-with-gdbs/what-is-an-srid.htm