

# Creating a Time-Series Animation with QGIS

Eric Xia, GIS & Data Assistant, Brown University Library

December 18, 2023

[https://libguides.brown.edu/gis\\_data\\_tutorials/wildlife](https://libguides.brown.edu/gis_data_tutorials/wildlife)

---

## Introduction

In this tutorial, we will demonstrate how to work with time-series data in QGIS with a wildlife tracks dataset to create an animation. We will walk through how to format the data, select time ranges with the *Temporal Control Panel*, and add a dynamic title bar. Finally, we will export images from QGIS and assemble them as a GIF with GIMP.

Not all data can or should be animated. A time-series animation should tell a story involving both time and space. For this to be effective, the data should have time-dependent variables which are also geographically related to one another. In this example, making an animation allows us to see two distinct gull migrations, following coastlines in the Pacific and Atlantic Oceans.

## 1 Timestamp Processing

### Formatting Text to Timestamp

A dataset has been prepared for the purposes of this tutorial. Download the zipped sample data from the tutorial website (link appears at top of this page). This data is sourced from Movebank, an online wildlife tracking database. Through the Argos satellite system, the locations of two gulls "CE" and "BX" have been recorded over the course of several years.<sup>1</sup>

We will be working with a comma-delimited (CSV) file with a text time field, which we will convert to a vector file with a QGIS Date & Time field. Before converting, we need to make sure that the CSV file contains an appropriate field for conversion. In order for the CSV to be compatible, the time field needs to be text in the following format: yyyy-mm-dd hh:mm:ss.

Open the CSV in a spreadsheet program. It might seem like we have an appropriate field for conversion, the *timestamp* column. However, take a close look at the data in the column. If we export this format to text, the data will not be correctly formatted: datetimes with single-digit hours, like 2017-07-15 0:03:29 will appear as NULL. We will have to use the following Excel<sup>2</sup> formula to convert the date/time column. The yyyy-mm-dd hh:mm:ss text format is the *only* way timestamps can be made into Date & Time fields in QGIS.

---

<sup>1</sup>Maftai, Mark. (2020). *Sabine's Gulls in the Juan de Fuca Eddy*. Movebank ID 304527187.

<sup>2</sup>Also works in Google Sheets and any other compatible editors.

=TEXT(C2,"yyyy-mm-dd hh:mm:ss")

Copy and paste this formula down in a new column. Then select the entire column, right click, copy and paste as special values to replace all the formulas with their output. Then, save the file as a CSV. Do not re-open the CSV in Excel; if you do, your local settings will likely re-set the formatting you just did. You can open the CSV in a text editor if you want to view it again.

In other cases, you might need to combine date and time values from two separate columns. Here's an Excel formula for combining date / time columns:

=CONCATENATE(TEXT(C2,"yyyy-mm-dd")," ",TEXT(D2,"hh:mm:ss"))

## Creating End Timestamps

The data now has the correct timestamps. However, right now there is no way for QGIS to know when to stop displaying a point. We know that at any particular moment, the individual (in this case the gull) is at a single position. That is, the end of a timestamp for an individual is the following timestamp for that individual. In order to create this end timestamp, we want to first sort the sheet first by individual, and then by timestamp.

This can be done in Excel by opening the *Sort* window, or in Google Sheets with *Data > Sort Range > Advanced Sorting Options*. In Google Sheets, check *Data has header row*. Then, sort by the individual (individual-local-identifier), and subsequently by the new timestamp column you created. Once the sheet is sorted, make a new column with header timestamp\_end. Set the first value to the second value of timestamp, and then copy the formula down the column. Finally, remove the final and boundary rows (the last row with BX as the individual).

Loc.	Time	Person	Loc.	Time	End	Person
...	1	A	...	1	2:30	A
...	2:30	A	...	2:30	5:30	A
...	5:30	A	...	5:30	1:30	A
...	1:30	B	...	1:30	2	B
...	2	B	...	2	3:30	B
...	3:30	B	...	3:30		B

We can infer end timestamps from start timestamps. Sorting by individual, and then time makes the next timestamp the correct end stamp, except for the boundary and last rows.

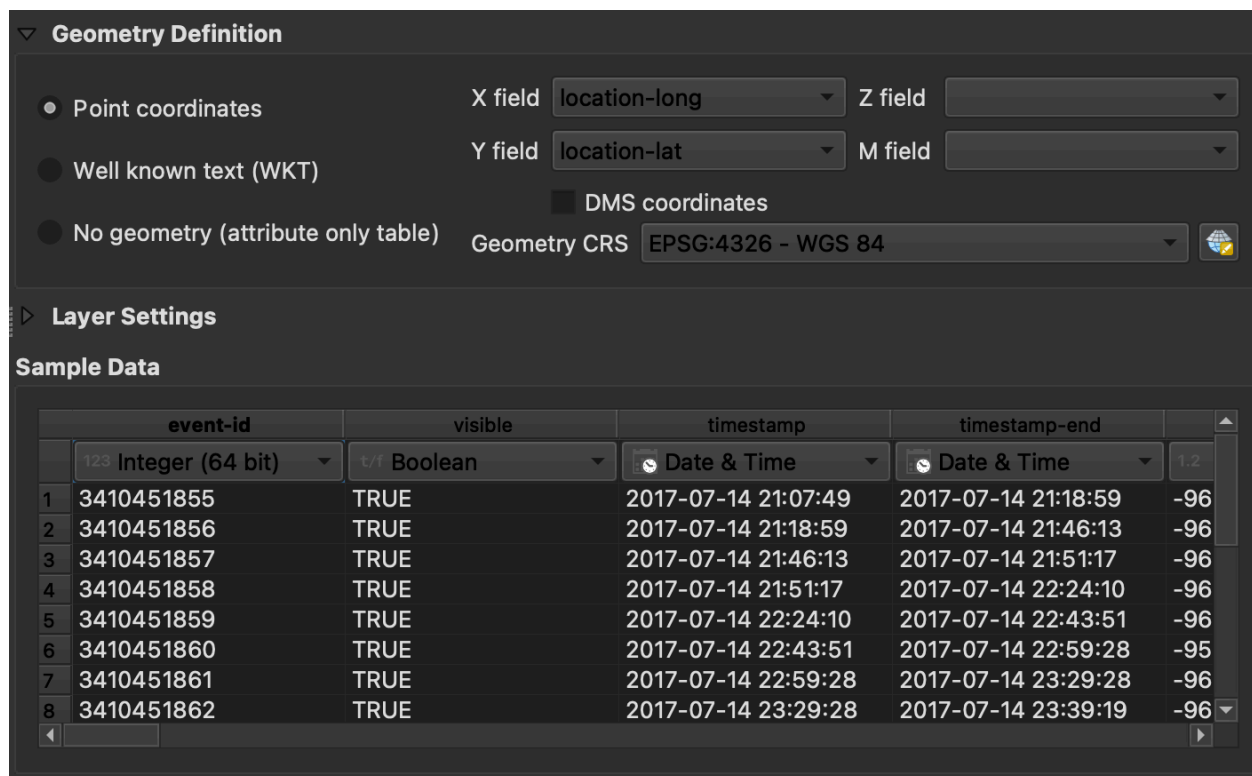
## 2 Create Animation in QGIS

### Convert CSV as a Vector File with Date

Once we've formatted the CSV, we can add it to QGIS and convert the datetime field.

Open a new, blank QGIS project. Click *Data Manager*, and select *Add Delimited Data*. Browse to the CSV file location. Under the import settings, specify the *X* field as the longitude (*location-long*), and the *Y* field as the latitude (*location-lat*). Click the dropdown under the timestamp field, and specify that the timestamps are in the *Date & Time* format. Do the same for the *timestamp\_end* field. If need be, also specify the *Geometry CRS* field. This is usually *EPSG:4326 - WGS84*, the standard coordinate reference system.

The *Geometry Definition* section should look like the following:



The screenshot shows the QGIS Data Manager interface. The **Geometry Definition** section is expanded, showing the following settings:

- Point coordinates** (selected): X field: location-long, Y field: location-lat, Z field: (empty), M field: (empty)
- Well known text (WKT)** (unselected): DMS coordinates: (unchecked)
- No geometry (attribute only table)** (unselected): Geometry CRS: EPSG:4326 - WGS 84

The **Layer Settings** section is also expanded, showing the **Sample Data** table:

	event-id	visible	timestamp	timestamp-end	
	Integer (64 bit)	Boolean	Date & Time	Date & Time	1.2
1	3410451855	TRUE	2017-07-14 21:07:49	2017-07-14 21:18:59	-96
2	3410451856	TRUE	2017-07-14 21:18:59	2017-07-14 21:46:13	-96
3	3410451857	TRUE	2017-07-14 21:46:13	2017-07-14 21:51:17	-96
4	3410451858	TRUE	2017-07-14 21:51:17	2017-07-14 22:24:10	-96
5	3410451859	TRUE	2017-07-14 22:24:10	2017-07-14 22:43:51	-96
6	3410451860	TRUE	2017-07-14 22:43:51	2017-07-14 22:59:28	-95
7	3410451861	TRUE	2017-07-14 22:59:28	2017-07-14 23:29:28	-96
8	3410451862	TRUE	2017-07-14 23:29:28	2017-07-14 23:39:19	-96

Now click *Add* to add the layer and plot the coordinates. If you want to verify the timestamp field was stored correctly, right click the layer and open the attribute table with *Open Attribute Table*. QGIS may have modified the formatting of the date based on your local settings, but as long as its stored as a *Date & Time* column things should work.

Once we have imported the layer as delimited text, we want to copy it to a richer data format. Right click the layer and select *Export > Save Features As*. Save it as a Geopackage, keeping *Add saved file to map* checked. Rename the newly saved layer gulls.

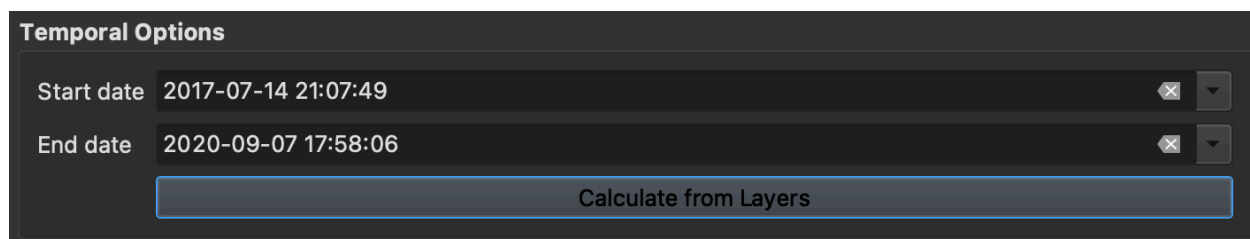
At this point, you can remove the original CSV layer, and add relevant vector data. For the purposes of this animation, we want relatively low detail vectors to display the coastlines of continents. Natural Earth has free and public domain 1:110M scale physical boundaries that will work for this animation (included in our sample data folder).



## Add Temporal Control

First, we will enable temporal control for the gulls layer. Double click the gulls layer to open the *Properties* tab, and select the *Temporal* tab. Check the *Dynamic Temporal Control* box. For *Configuration*, choose *Separate Fields for Start and End Date / Time*, and for *Limits Include Start / Exclude End*. Choose the *timestamp* field for the start field, and the *timestamp-end* field for the end field. Click *Apply*, and then *OK*.


Now we will set the project time range. Go to *Project > Properties > Temporal*. Hit the *Calculate from Layers* button to set the date range for the project. If only an end date appears and no start date, make sure that you've saved the layer as a GeoPackage.

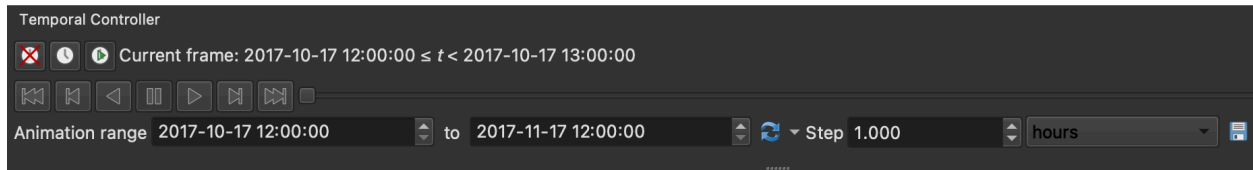


Temporal Options	
Start date	2017-07-14 21:07:49
End date	2020-09-07 17:58:06
<b>Calculate from Layers</b>	

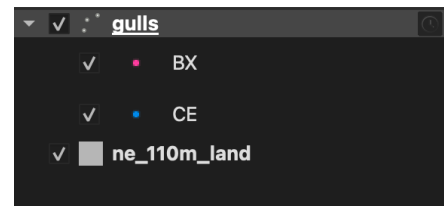
## Animate Time Series

With temporal control enabled, we can animate the time series from the *Temporal Control Panel*. Right click on a blank area of the toolbars, and check the *Temporal Control Panel* to open it.


In the control panel, click *Animated Temporal Navigation* . In the *Animation range*, select 2017-10-17 to 2017-11-17. Input a step of four hours: this will display all observed locations within a four hour time span. Press play or move the slider to see the animation.



At this point, you can adjust the display options to differentiate individuals. Double click the gulls layer to open the layer properties. Then, under the *Symbology* tab, change to the symbol type to *Categorized*, selecting *individual-local-identifier* as the value. Then, click the *Classify* button; color classes for BX and CE should pop up. Click *Apply*, and then *OK*. Our layers should be similar to the right.



## Add Time Label

Our data is all in place, but its useful to have the time display somewhere for reference. We can add a label that dynamically changes based on the current map frame. Access the decoration label window by going to *View > Decorations > Title Label* . Check *Enable Title Label*, and then enter into the text box:

```
[%format_date(@map_start_time, 'MM-dd-yyyy')%]
```

This displays the current map's start time variable in a readable format. You can set the label display however you want. We've set the title label placement to bottom left, changed the font size to 25pt, and made the title box transparent.

## Export Time Series

Now we will save animation frames as PNGs, which can be assembled into a GIF.

Click the yellow gear wheel at the end of the *Temporal Control Panel*. The frame rate is a bit of a misnomer, as QGIS is simply exporting images, not making a GIF. A frame rate of 1 works for the gulls dataset. Later on, you can set the frame rate when assembling the GIF.

Now, click the save button on the *Temporal Control Panel* (*\*not\** the save button on the main toolbar). The settings here determine the map extent to be exported. *Calculate from Layer* sets the extent of the animation from the time-dependent layer, which will stretch the extent to fit all of the points. *Draw on Canvas* is another useful feature which lets you set a custom extent. For the tutorial, *Draw on Canvas* is used to clip the map to the following region:



10-18-2017

Make sure to specify an output directory. Depending on the number of steps and the time scale, you may end up generating lots of image files. With the timespan specified of one month and 4 hour frames, we generate 185 images.

### 3 Assemble GIF

#### Create GIF with GIMP

Finally, we can assemble the GIF from our directory of images. There are many ways to do this. A straightforward method is to use the free and open source image software GIMP, available at [gimp.org](http://gimp.org).

To make an animation in GIMP, it's necessary to load each image as a layer, which can take a long time when dealing with hundreds of images. To avoid this, we can keep every tenth image: search for `*1.png` and copy those files to another folder. Open GIMP, and select *File > Open as Layers*. Choose the images for the GIF. The images will stack on each other: the GIF will display from the bottom layer up, so the last image should be on top.

Now, select *Filters > Animation > Optimize (for GIF)*. Once the GIF is done optimizing, you can play it back with *Filters > Animation > Playback*. To export the GIF, select *File > Export As*. Open the Select File Type dropdown and select GIF. Name your new GIF. Finally, an *Export Image as GIF* window should open up. Check the *As Animation* box, and then *Export*. Your GIF should be exported to the location specified. Note that you can modify the speed at which frames move in this dialog box.

## Create GIF with Python

Alternatively, you can run the following Python script outside of the output directory, in this case gulls-png. It requires an environment with the imageio and os packages installed. The script reads images sequentially, and writes them to a GIF file.

```
import imageio
import os

images = []

for root,dir, filenames in os.walk('gulls-png'):
    ids = [filename[-8:-4] for filename in filenames]
    TOTAL = int(max(ids))
    for num in range(0,TOTAL):
        path = f'gulls_walkthrough{num:04}.png'
        fn = os.path.join(root, path)
        if(os.path.isfile(fn)):
            images.append(imageio.imread(fn))

imageio.mimsave('gulls.gif', images, fps=60)
```