Instructor Solution - Project 04 (Geographic Animation)

This document has an example solution to the geographic animation project. It's meant to show my thought process so you can see how I built the script and interacted with MATLAB Copilot.

As a reminder, here is the project description:

Objective: Animate a geographic plot to show movement over time using GPS or location data.

Requirements:

- A base map or geographic axes using geoplot, geoscatter, or similar.
- Animated movement of one or more points over time.
- A static set of axes limits so the map isn't zooming in or out during the animation
- Optional: Add contextual elements like text labels that also change over time.

Table of Contents

Step 1: Load and inspect the dataset....1
Step 2: Construct a prompt for Copilot Chat....2
Step 3: Address an Error....4
    Get an Explanation....4
Step 4: Fix the basemap....7
Step 5: Check requirements....9
Step 6: Extra customizations....9
  Increase latitude limits....9
  Make the marker a filled-in circle....9
  Change the way the line updates....10
  Decrease pause length....10
  Add a hold off....10
Step 7: Add a description in a comment....10
Final Animation....11

Step 1: Load and inspect the dataset

I used the cycloneTracks data from this doc example. To obtain the data, I ran the command:

```
openExample('graphics/GeoplotTableDataExample')
```

This downloads a data file with the location data. I copied the file `cycloneTracks.mat` into a new folder and used some of the code in the example to isolate a single cyclone track.

```
load cycloneTracks.mat cycloneTracks
singleTrack = cycloneTracks(cycloneTracks.ID == 1320,:)
```

```
singleTrack = 40×8 table
```

...

	ID	Name	Time	Grade	Latitude	Longitude	Pressure
1	1320	"PABUK"	19-Sep-2013 06:00:00	2	14	148.5000	1004
2	1320	"PABUK"	19-Sep-2013 12:00:00	2	14.9000	149	1002

	ID	Name	Time	Grade	Latitude	Longitude	Pressure
3	1320	"PABUK"	19-Sep-2013 18:00:00	2	15.8000	149.3000	1002
4	1320	"PABUK"	20-Sep-2013 00:00:00	2	16.9000	149.7000	1002
5	1320	"PABUK"	20-Sep-2013 06:00:00	2	18.6000	149.5000	1002
6	1320	"PABUK"	20-Sep-2013 12:00:00	2	19.3000	148.2000	1002
7	1320	"PABUK"	20-Sep-2013 18:00:00	2	19.7000	146.8000	1000
8	1320	"PABUK"	21-Sep-2013 00:00:00	2	19.7000	145.9000	998
9	1320	"PABUK"	21-Sep-2013 06:00:00	3	19.8000	145.2000	996
10	1320	"PABUK"	21-Sep-2013 12:00:00	3	20	144.7000	992
11	1320	"PABUK"	21-Sep-2013 18:00:00	3	20.6000	144.1000	990
12	1320	"PABUK"	22-Sep-2013 00:00:00	3	21.3000	143.3000	990
13	1320	"PABUK"	22-Sep-2013 06:00:00	4	22	142.7000	985
14	1320	"PABUK"	22-Sep-2013 12:00:00	4	22.5000	141.9000	980

⋮

This table preview in the Live Editor is helpful to inspect the variable names. Another way to get a list of the variable names is using the summary function:

```
summary(singleTrack)
```

```
singleTrack: 40×8 table

Variables:

    ID: double
    Name: string
    Time: datetime
    Grade: double
    Latitude: double
    Longitude: double
    Pressure: double
    WindSpeed: double

Statistics for applicable variables:

                 NumMissing              Min                        Median                  Max

    ID               0                           1320                        1320               1320
    Name             0
    Time             0           19-Sep-2013 06:00:00        24-Sep-2013 03:00:00        29-Sep-2013        24-Sep-2
    Grade            0                              2                           4                  6
    Latitude         0                             14                          26            49.6000
    Longitude        0                       138.7000                    145.2000           181.5000
    Pressure         0                            965                    984.5000               1012
    WindSpeed        0                              0                     42.5000                 60
```

The variable names can be copied into a prompt and used by Copilot to help generate code.

Step 2: Construct a prompt for Copilot Chat

Next, I wrote a prompt for Copilot Chat. I decided to start with Copilot Chat for two reasons:

- I wanted to use Chat's ability to search the documentation.
- I wanted a conversation in the chat so I could ask follow-up questions.

For the prompt, I tried to avoid the pitfall of "asking for too much as once" by only including some of the requirements in this first prompt. I also included the variables in `singleTrack` to provide context. Here is the full prompt:

```
I'd like you to animate a geographic plot to show movement over time using location data. The plot
should have:

- A base map or geographic axes using geoplot, geoscatter, or similar.
- Animated movement of one or more points over time.

The dataset I'm using is cyclone track data. I've already extracted a single cyclone's track into a
variable named singleTrack.
Here's the variables in the table:

    ID: double
    Name: string
    Time: datetime
    Grade: double
    Latitude: double
    Longitude: double
    Pressure: double
    WindSpeed: double
```

The response I received had the following block of code:

```
% Assuming singleTrack is a table with the relevant variables
% singleTrack should contain columns: Time, Latitude, Longitude

% Create a figure for the animation
figure;
geobasemap streets; % Set the basemap
hold on;

% Plot the initial position of the cyclone
h = geoplot(singleTrack.Latitude(1), singleTrack.Longitude(1), 'ro', 'MarkerSize', 10);

% Set geographic limits based on the data
geolimits([min(singleTrack.Latitude) max(singleTrack.Latitude)], ...
           [min(singleTrack.Longitude) max(singleTrack.Longitude)]);

% Loop through each time step to update the plot
for i = 1:height(singleTrack)
    % Update the position of the cyclone
    set(h, 'Latitude', singleTrack.Latitude(i), 'Longitude', singleTrack.Longitude(i));

    % Optionally, you can plot the path of the cyclone
    geoplot(singleTrack.Latitude(1:i), singleTrack.Longitude(1:i), 'b-');

    % Pause to create an animation effect
    pause(0.5); % Adjust the pause duration as needed
end
```

```
title('Cyclone Track Animation');
```

This is a promising start but try running the code. You'll see that it produces an error:

```
Error using matlab.graphics.chart.primitive.Line/set Unrecognized property Latitude for class Line.
```

Step 3: Address an Error

Get an Explanation

To understand the issue, I started by clicking **Explain Error.** One of the suggestions was to use the command **`get(h)`** to list the properties of the `Line` object. The output was a long list of properties, several of which started with "Longitude" and "Latitude". I wasn't sure if Copilot could pick the right one, but I decided to try before visiting the documentation myself.

Here's the follow-up prompt I asked in Copilot Chat:

```
Here's the output of get(h). What property should I by setting instead?

       AffectAutoLimits: on
     AlignVertexCenters: off
             Annotation: [1×1 matlab.graphics.eventdata.Annotation]
           BeingDeleted: off
             BusyAction: 'queue'
          ButtonDownFcn: ''
               Children: [0×0 GraphicsPlaceholder]
               Clipping: on
                  Color: [1 0 0]
              ColorMode: 'manual'
            ContextMenu: [0×0 GraphicsPlaceholder]
              CreateFcn: ''
        DataTipTemplate: [1×1 matlab.graphics.datatip.DataTipTemplate]
              DeleteFcn: ''
            DisplayName: ''
       HandleVisibility: 'on'
                HitTest: on
          Interruptible: on
           LatitudeData: 14
       LatitudeDataMode: 'manual'
     LatitudeDataSource: ''
       LatitudeVariable: ''
               LineJoin: 'round'
              LineStyle: 'none'
          LineStyleMode: 'manual'
              LineWidth: 0.5000
          LongitudeData: 148.5000
      LongitudeDataMode: 'manual'
    LongitudeDataSource: ''
      LongitudeVariable: ''
                 Marker: 'o'
        MarkerEdgeColor: 'auto'
        MarkerFaceColor: 'none'
          MarkerIndices: 1
             MarkerMode: 'manual'
             MarkerSize: 10
```

```
                Parent: [1×1 GeographicAxes]
         PickableParts: 'visible'
              Selected: off
    SelectionHighlight: on
           SeriesIndex: 1
           SourceTable: [0×0 table]
                   Tag: ''
                  Type: 'line'
              UserData: []
               Visible: on
                 XData: [1×0 double]
             XDataMode: 'manual'
           XDataSource: ''
             XVariable: ''
                 YData: [1×0 double]
             YDataMode: 'manual'
           YDataSource: ''
             YVariable: ''
                 ZData: [1×0 double]
             ZDataMode: 'auto'
           ZDataSource: ''
             ZVariable: ''
```

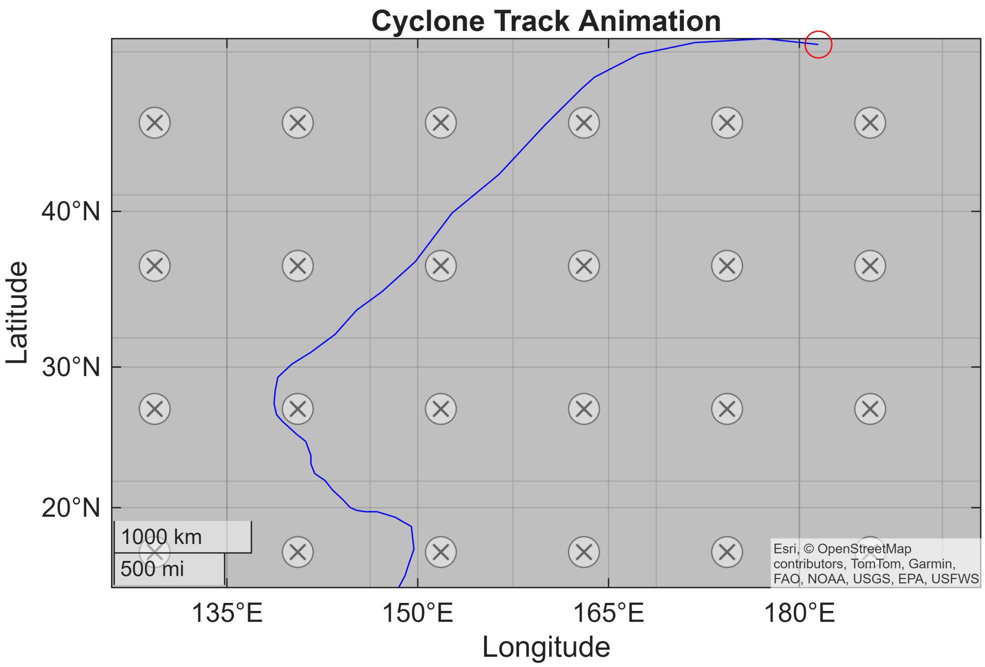
I wasn't sure if this was going to work, but thankfully, Copilot was able to identify the right properties: `LatitudeData` and `LongitudeData`. Here is the output from Copilot:

```
The error you encountered arises because the Line object created by geoplot does
not have properties named Latitude and Longitude. Instead, you should use the
LatitudeData and LongitudeData properties to update the position of the plotted point.

Here's how you can modify your code to correctly update the geographic plot:

% Create a figure for the animation
figure;
geobasemap streets; % Set the basemap
hold on;

% Plot the initial position of the cyclone
h = geoplot(singleTrack.Latitude(1), singleTrack.Longitude(1), 'ro', 'MarkerSize', 10);

% Set geographic limits based on the data
geolimits([min(singleTrack.Latitude) max(singleTrack.Latitude)], ...
          [min(singleTrack.Longitude) max(singleTrack.Longitude)]);

% Loop through each time step to update the plot
for i = 1:height(singleTrack)
    % Update the position of the cyclone using LatitudeData and LongitudeData
    h.LatitudeData = singleTrack.Latitude(i);
    h.LongitudeData = singleTrack.Longitude(i);

    % Optionally, you can plot the path of the cyclone
    geoplot(singleTrack.Latitude(1:i), singleTrack.Longitude(1:i), 'b-');

    % Pause to create an animation effect
    pause(0.5); % Adjust the pause duration as needed
end
```

```
title('Cyclone Track Animation');

Key Changes:
 - Use h.LatitudeData and h.LongitudeData to update the position of the point on the geographic plot.
 - This approach ensures that you are modifying the correct properties of the Line object created by
geoplot.
This should resolve the error and allow you to animate the cyclone's movement correctly.
```

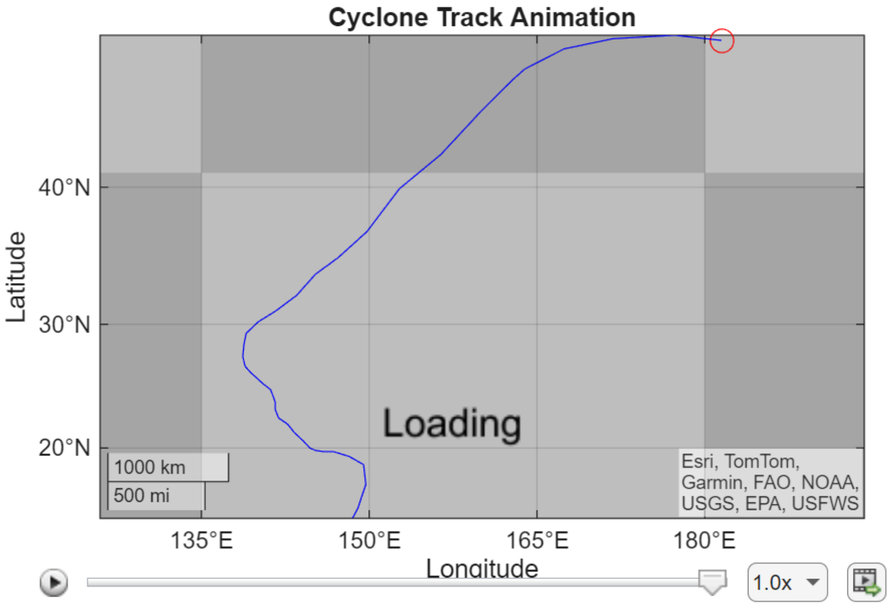
I copied the code suggestion into my script and ran the code

```matlab
% Create a figure for the animation
figure;
geobasemap streets; % Set the basemap
hold on;

% Plot the initial position of the cyclone
h = geoplot(singleTrack.Latitude(1), singleTrack.Longitude(1), 'ro', 'MarkerSize',
10);

% Set geographic limits based on the data
geolimits([min(singleTrack.Latitude) max(singleTrack.Latitude)], ...
           [min(singleTrack.Longitude) max(singleTrack.Longitude)]);

% Loop through each time step to update the plot
for i = 1:height(singleTrack)
    % Update the position of the cyclone using LatitudeData and LongitudeData
    h.LatitudeData = singleTrack.Latitude(i);
    h.LongitudeData = singleTrack.Longitude(i);

    % Optionally, you can plot the path of the cyclone
    geoplot(singleTrack.Latitude(1:i), singleTrack.Longitude(1:i), 'b-');

    % Pause to create an animation effect
    pause(0.5); % Adjust the pause duration as needed
end

title('Cyclone Track Animation');
```



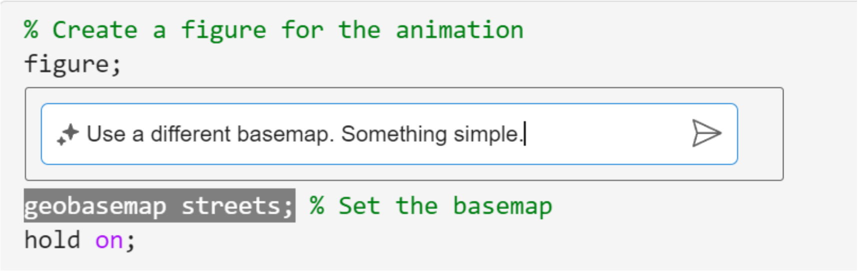
Step 4: Fix the basemap

The above code successfully created an animation of a dot moving across a set of axes. However, there was no visible map. Here's a screenshot from my animation:



The background didn't load. I've made geographic plots before, so I knew the issue was with the **`basemap`** command. If I didn't know, though, I would've asked Copilot Chat a question like "Why do I see a marker and line, but no background map? It's just a gray image that says `loading`". The response from Copilot will point you towards changing the basemap.

Once I decided to change the basemap, I used **Ask Copilot** to change the code. I wrote a fairly simple prompt: "Use a different basemap. Something simple"



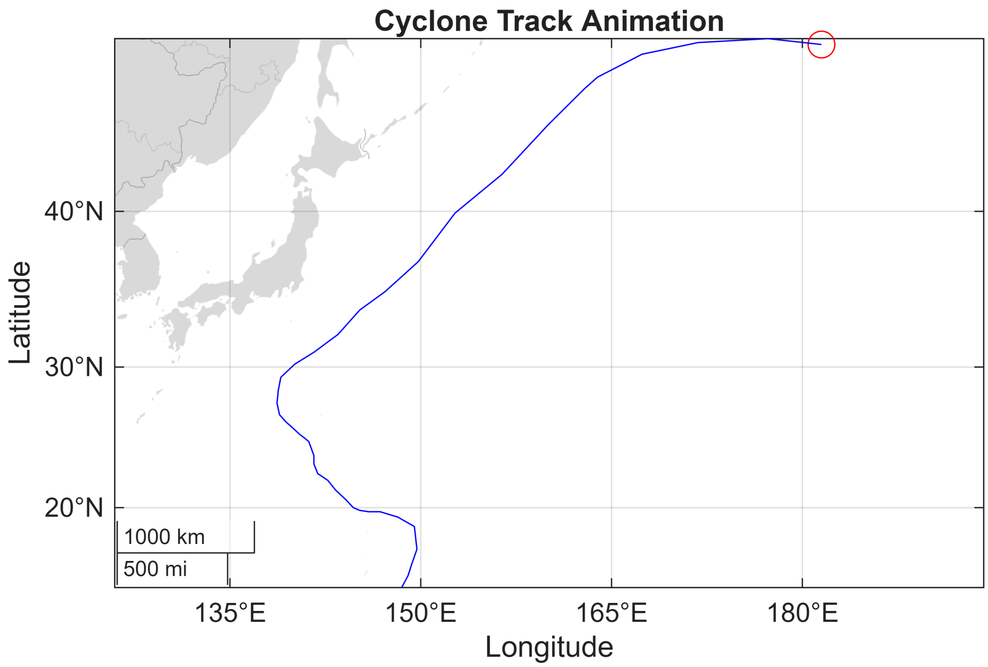
```
% Create a figure for the animation
figure;
```

Use a different basemap. Something simple.

```
geobasemap streets; % Set the basemap
hold on;
```

It suggested `basemap grayland`, which revealed the cyclone was near the coast of Japan.

```
% Create a figure for the animation
figure;
geobasemap grayland; % Set a simple basemap
hold on;

% Plot the initial position of the cyclone
h = geoplot(singleTrack.Latitude(1), singleTrack.Longitude(1), 'ro', 'MarkerSize',
10);

% Set geographic limits based on the data
geolimits([min(singleTrack.Latitude) max(singleTrack.Latitude)], ...
           [min(singleTrack.Longitude) max(singleTrack.Longitude)]);

% Loop through each time step to update the plot
for i = 1:height(singleTrack)
    % Update the position of the cyclone using LatitudeData and LongitudeData
    h.LatitudeData = singleTrack.Latitude(i);
    h.LongitudeData = singleTrack.Longitude(i);

    % Optionally, you can plot the path of the cyclone
    geoplot(singleTrack.Latitude(1:i), singleTrack.Longitude(1:i), 'b-');

    % Pause to create an animation effect
    pause(0.5); % Adjust the pause duration as needed
end

title('Cyclone Track Animation');
```



Step 5: Check requirements

At this point the animation is essentially complete. It fits the three main requirements

- A base map or geographic axes using geoplot, geoscatter, or similar. ✅
- Animated movement of one or more points over time. ✅
- A static set of axes limits so the map isn't zooming in or out during the animation ✅

Note that the code met the last requirement without prompting due to the function `geolimits`. Had Copilot *not* included this, I would've used Copilot Chat to get a suggestion.

Step 6: Extra customizations

There were a few things I changed manually to improve the plot. This is something I often do with Copilot suggestions. It provides a great first draft, but then I tweak the result to my liking.

Increase latitude limits

I didn't like how the cyclone path was touching the bounding box. Therefore, I manually added a couple extra degrees of latitude to the limits in order to improve the appearance.

```
% Set geographic limits based on the data (and some padding)
geolimits([min(singleTrack.Latitude)-2 max(singleTrack.Latitude)+2], ...
          [min(singleTrack.Longitude) max(singleTrack.Longitude)]);
```

Make the marker a filled-in circle

I wanted the marker to stand out more in the animation, so I made it a filled-in circle. From my years of MATLAB experience, I knew that the property I needed to change was `MarkerFaceColor`, but this is something that Copilot can help with if you didn't know that.

```
h1 = geoplot(singleTrack.Latitude(1), singleTrack.Longitude(1), 'ro', ...
    'MarkerSize', 6, 'MarkerFaceColor', 'r' );
```

Change the way the line updates

Notice in the Copilot suggestion, the red circle is updated by changing properties, but the blue line is newly created in each loop iteration. I decided to change the blue line to update the same way as the red circle. This makes the code more efficient, but it's not strictly necessary.

This change requires creating the line object before the loop and then changing the update code inside the loop. Notice I also switched the order to make the marker last. This ensures the marker appears on top of the line.

```
% Plot the initial position of the cyclone with a line and circle
h1 = geoplot(singleTrack.Latitude(1), singleTrack.Longitude(1), 'b-');
h2 = geoplot(singleTrack.Latitude(1), singleTrack.Longitude(1), 'ro', ...
    'MarkerSize', 6, 'MarkerFaceColor', 'r' );

% Loop through each time step to update the plot
for i = 1:height(singleTrack)

    % Update the position of the line using LatitudeData and LongitudeData
    h1.LatitudeData = singleTrack.Latitude(1:i);
    h1.LongitudeData = singleTrack.Longitude(1:i);

    % Update the position of the circle
    h2.LatitudeData = singleTrack.Latitude(i);
    h2.LongitudeData = singleTrack.Longitude(i);

    % Pause to create an animation effect
    pause(0.5); % Adjust the pause duration as needed
end
```

Decrease pause length

A pause of 0.5 seconds isn't necessary for this simple animation, so I decreased it to 0.1 seconds

```
pause(0.1); % Adjust the pause duration as needed
```

Add a hold off

The Copilot suggestion uses `hold on` to add multiple objects to a single plot, but it doesn't turn the hold off. I always try to do this so I don't screw up any other plots in a script. Thus, I added the following line to the end of the script:

```
hold off;
```

Step 7: Add a description in a comment

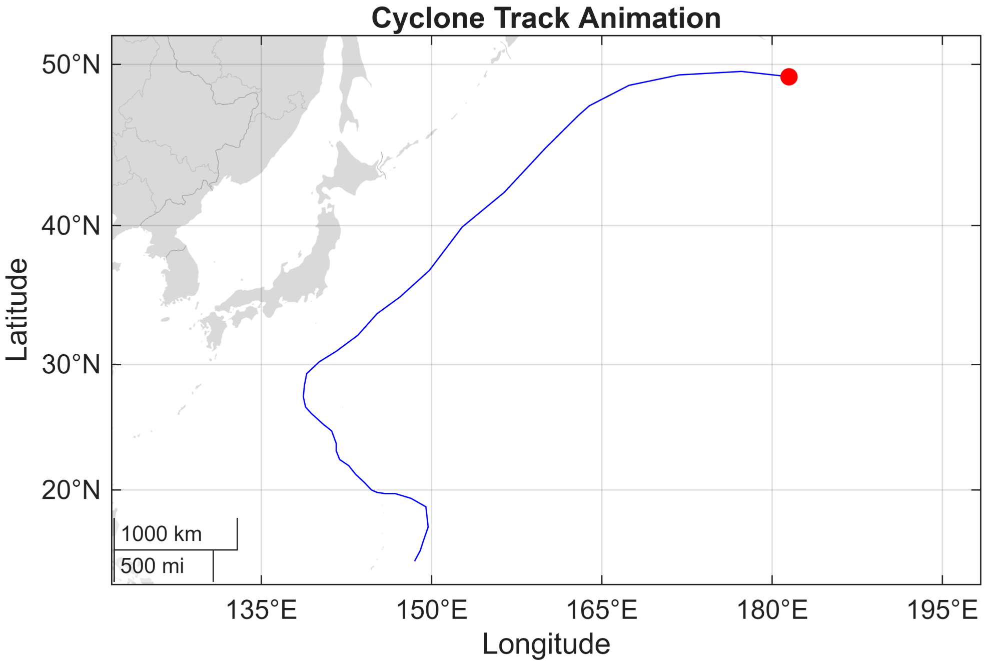
One of the general requirements was a short description of the script's code. I used **Ask Copilot** for this task with the prompt, "Add a short description to this script as a comment". Here is the output:

```
% This code creates an animated plot of the cyclone's track over time. It
% initializes a figure with a specified basemap, plots the initial
% position, sets geographic limits, and updates the plot in a loop to show
% the cyclone's movement.
```

Final Animation

Here is the final script:

```
% This code creates an animated plot of the cyclone's track over time. It
% initializes a figure with a specified basemap, plots the initial
% position, sets geographic limits, and updates the plot in a loop to show
% the cyclone's movement.

% Load the data
load cycloneTracks.mat cycloneTracks
singleTrack = cycloneTracks(cycloneTracks.ID == 1320,:);

% Create a figure for the animation
figure;
geobasemap grayland; % Set a simple basemap
hold on;

% Set geographic limits based on the data (and some padding)
geolimits([min(singleTrack.Latitude)-2 max(singleTrack.Latitude)+2], ...
          [min(singleTrack.Longitude) max(singleTrack.Longitude)]);

% Plot the initial position of the cyclone with a line and circle
h1 = geoplot(singleTrack.Latitude(1), singleTrack.Longitude(1), 'b-');
h2 = geoplot(singleTrack.Latitude(1), singleTrack.Longitude(1), 'ro', ...
    'MarkerSize', 6, 'MarkerFaceColor', 'r' );
title('Cyclone Track Animation');

% Loop through each time step to update the plot
for i = 1:height(singleTrack)

    % Update the position of the line using LatitudeData and LongitudeData
    h1.LatitudeData = singleTrack.Latitude(1:i);
    h1.LongitudeData = singleTrack.Longitude(1:i);

    % Update the position of the circle
    h2.LatitudeData = singleTrack.Latitude(i);
    h2.LongitudeData = singleTrack.Longitude(i);

    % Pause to create an animation effect
    pause(0.1); % Adjust the pause duration as needed
end

hold off;
```


Cyclone Track Animation
50°N
40°N
30°N
20°N
Latitude
1000 km
500 mi
135°E
150°E
165°E
180°E
195°E
Longitude