Scientific and Information Visualization

Dyuthi Vivek
IMT2022523
IIIT Bangalore
Bangalore, India
Dyuthi.Vivek@iiitb.ac.in

Saniya Ismail Kondkar
IMT2022128
IIIT Bangalore
Bangalore, India
Saniya.Ismail@iiitb.ac.in

Ragini Metlapalli
IMT2022029
IIIT Bangalore
Bangalore, India
Metlapalli.Ragini@iiitb.ac.in

I. Scientific Visualization

The months of August, September, and October 2021, along with the selected dates—August 26, August 29, September 1, September 4, and September 10—were specifically chosen to capture and visualize the progression and aftermath of Hurricane Ida [1]. Hurricane Ida was one of the most destructive storms of the 2021 Atlantic hurricane season, impacting regions across the Gulf Coast, and later moving through the eastern United States. Ida intensified rapidly after forming in late August, becoming a powerful Category 4 hurricane.

The chosen dates provide a timeline of Ida's development, peak intensity, and dissipation. The first date, August 26, represents Ida's early formation phase, while August 29 captures the storm at its peak during landfall. Subsequent dates in early September, including September 1 and September 4, allow us to observe Ida's transition as it moved inland, weakened, and continued to bring widespread rainfall and flooding across the northeastern United States. Finally, data from September 10 gives us a sense of post-storm conditions.

The path followed by the hurricane Ida has been shown in the Fig 1.

Fig 1: The path of hurricane Ida (2021). Image courtesy: [2]

The meteorological data used for the scientific visualization plots has been taken from the gridMET dataset [3].

A. Quiver Plot

1) Implemenation: To implement the quiver plot for visualizing wind patterns, the `vs` (wind speed at 10m) and `th` (wind direction at 10m) netCDF files were used. The Python modules: xarray [4], netCDF4 [5] and pandas [6] were used to load data from a NetCDF (nc) file and convert it into a DataFrame for more straightforward data handling and processing. For visualizing these wind patterns, cartopy [7] was utilized alongside matplotlib [8] to generate georeferenced plots. Cartopy provided essential geographic context, adding coastlines and accurately projecting wind vectors over maps. Additionally, imageio [9] was used to compile individual plot images into a GIF.

2) Experiments: Grid sampling experiments were performed using meshgrid sampling and simple subsampling.

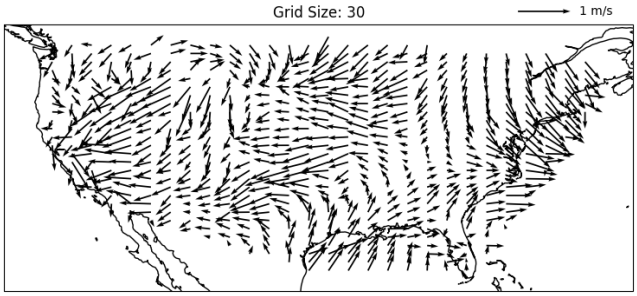


Fig 1.1.1: Quiver plot with grid sampling, with grid size = 30 x 30

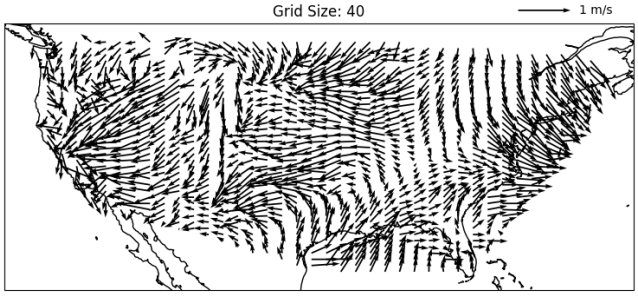


Fig 1.1.2: Quiver plot with grid sampling, with grid size = 40 x 40

Meshgrid sampling is a technique used to create a structured, uniformly spaced grid over a given spatial range, enabling consistent sampling of data points. It does interpolation of the wind vectors onto a defined grid, helping to visualize patterns across the map with evenly distributed arrows. By setting specific intervals for latitude and longitude, meshgrid sampling helps capture regional variations in data—like wind direction and speed in different map areas—without missing

critical shifts that might occur in smaller regions, such as those influenced by hurricanes. Scipy [10] was used for the interpolation. In Fig 1.1.1, the grid size is 30 x 30 i.e, the latitude and longitude ranges are divided into a 30 x 30 grid. Fig 1.1.2 uses a grid size of 40 x 40. A grid size of 30 x 30 was chosen as it ensured that the arrows in the map were not overcrowded.

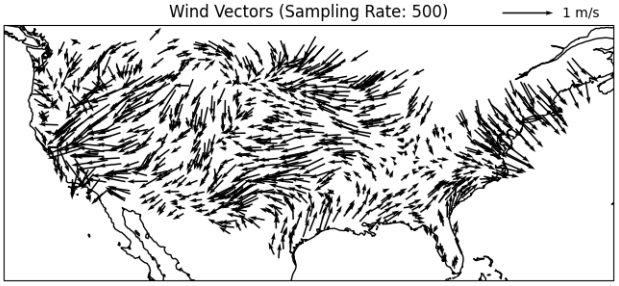


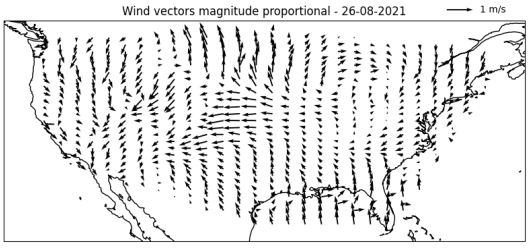
Fig 1.1.3: Quiver plot with simple subsampling: every 500th data point was sampled

Simple subsampling involved selecting points at regular intervals. In Fig 1.1.3, the quiver plot was created by sampling every 500th data point. The method is efficient but may not capture spatially variable wind patterns adequately, especially in areas with rapidly changing winds like hurricane centers. Therefore, meshgrid sampling was chosen for the plots, for its consistent spatial coverage across the map.

2 quiver plots were drawn for each of the sampled dates: with magnitude proportional vectors, and with same sized vectors coloured according to magnitude. The viridis colourmap was chosen as the colours in the colour scheme are perceptually uniform, and it is accessible to people with color vision deficiencies.

The magnitude-proportional vectors provide an intuitive representation of wind strength by varying the length of the vectors in accordance with their magnitude, making it easy to compare wind speeds at a glance. However, this approach can sometimes make it challenging to perceive wind direction uniformly if vector lengths vary significantly. In contrast, using same-sized vectors colored according to magnitude with the perceptually uniform viridis colourmap ensures that wind direction remains consistently represented while wind speed differences are visually communicated through color intensity. But, it may be less immediate in conveying magnitude differences compared to varying vector lengths.

3) Inferences:



:

Fig 1.1.4: Quiver plot for 26-08-2021 with magnitude proportional vectors

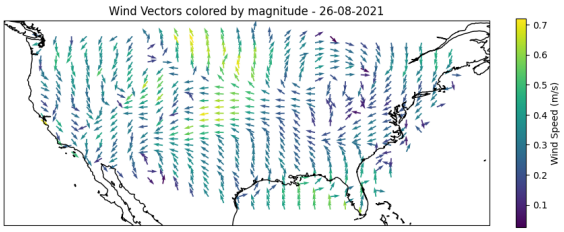


Fig 1.1.5: Quiver plot for 26-08-2021 with same sized vectors

In the plot from August 26, we see a more uniform wind pattern across the country, with no significant central disturbance in the Gulf region. This suggests that Ida had not yet intensified or reached a stage where its effects were noticeable in these wind vectors across the U.S.

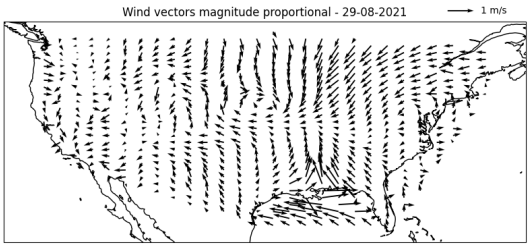


Fig 1.1.6: Quiver plot for 29-08-2021 with magnitude proportional vectors

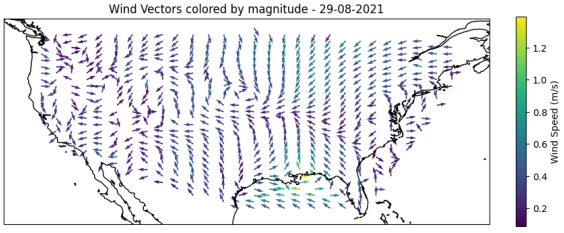


Fig 1.1.7: Quiver plot for 29-08-2021 with same sized vectors

By August 29, however, we observe a concentrated area of strong, converging wind vectors around the southeastern U.S., specifically near Louisiana and surrounding states. This area of intensified, inward-pointing vectors indicates a low-pressure system, characteristic of hurricanes. This pattern is a clear signature of Hurricane Ida making landfall and influencing wind patterns with higher speeds and a converging flow.

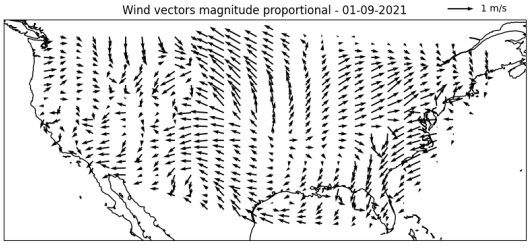


Fig 1.1.8: Quiver plot for 01-09-2021 with magnitude proportional vectors

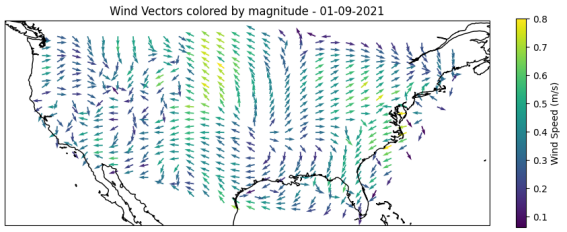


Fig 1.1.9: Quiver plot for 01-09-2021 with same sized vectors

In this quiver plot for September 1, 2021, we see the progression of Hurricane Ida's influence as it moves inland from the Gulf region toward the northeastern United States. The wind vectors show a strong flow pattern extending from the southeastern U.S. up towards the northeast. This reflects the remnants of Ida moving inland, consistent with the typical path of hurricanes after landfall. Unlike the August 29 plot, where the wind vectors were strongly converging around Louisiana, this plot shows a more dispersed pattern, indicating that the hurricane has weakened and spread out as it moved further inland. There is noticeable wind activity near the northeastern states, showing the trajectory of the storm as it brings heavy rain, wind, and potential flooding to that area.

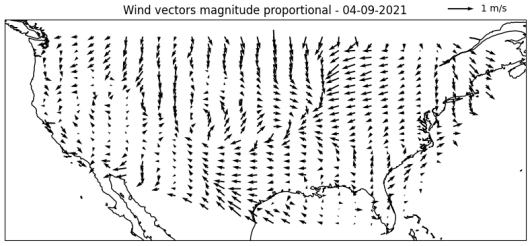


Fig 1.1.10: Quiver plot for 04-09-2021 with magnitude proportional vectors

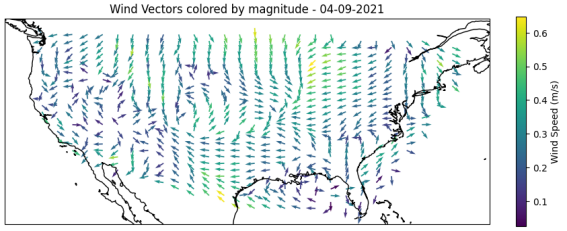


Fig 1.1.11: Quiver plot for 04-09-2021 with same sized vectors

On September 4, 2021, the wind pattern is still notable, with vectors concentrated over the eastern U.S., but there is a visible decrease in density and intensity. This likely indicates that the remnants of Ida have dissipated somewhat but are still causing breezy conditions as they move further northeast.

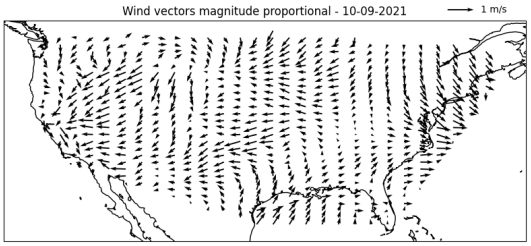


Fig 1.1.12: Quiver plot for 10-09-2021 with magnitude proportional vectors

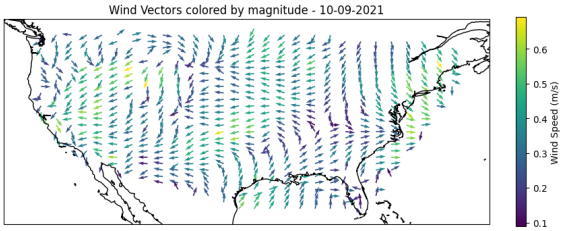


Fig 1.1.13: Quiver plot for 10-09-2021 with same sized vectors

By September 10, 2021, the wind vectors are more dispersed and spread uniformly across the country, indicating that Ida's direct influence has subsided. The wind patterns have returned to a more typical, less concentrated distribution, with no significant indication of high storm-related winds.

B. Color mapping

1) Implementation: For implementing color maps, climate data for precipitation and maximum air temperature from 2021 was loaded from NetCDF files. Essential libraries like netCDF4 [5] for NetCDF file handling, pandas [6] for data management, numpy [14] for numerical operations, and for visualization, matplotlib [8] and the cartopy [7] library for effective geographic plotting were used. Cartopy allowed setting map projections, adding coastlines and borders, and adding precise longitude and latitude ticks. And the imageio [9] library was used to generate GIFs.

Different color scales and palettes have been experimented with, and it was found that each climate variable—precipitation, temperature—has different characteristics that require different color mapping styles to bring out their unique patterns.

Using the selected color scale, palette, etc., color map plots for the selected dates (August 26, August 29, September 1, September 4, and September 10) as sampled by us have been plotted.

2) Parametric Mapping: To ensure consistent comparisons across different dates, the approach involved calculating global minimum and maximum values for each climate variable (precipitation and temperature) across the entire dataset. This method of setting fixed parametric bounds, based on global minima and maxima rather than local daily extremes, avoids shifts in color intensity that could distort data interpretation over time.

The precipitation values ranged from 0 mm (no rainfall) to a maximum of 504.2 mm, while temperature ranged from 243.2 K (cooler areas) to 327.0 K (warmest regions). Using these global values as color map bounds across all timesteps ensures that similar values are represented consistently in every plot, making it possible to observe true variations in climate impact without misleading fluctuations in color intensity. Without this standardization, each day's plot might adjust to its own minimum and maximum values, resulting in inconsistent color scaling and making it harder to directly compare data across dates.

3) Choice of Color Mapping, Palette, and Scale:

a) Precipitation: Among all the color palettes that have been tried—coolwarm (Fig 1.2.1), YlGnBu (Fig 1.2.2), and Viridis (Fig 1.2.3), for precipitation, divergent color maps (e.g., coolwarm) are less appropriate as precipitation data does not have an inherent midpoint, and extreme values should ideally go in one direction (from dry to heavy rainfall). The sequential color map, specifically YlGnBu (Yellow-Green-Blue), worked well. This palette moves from light yellow (representing minimal rainfall) to dark blue (indicating heavy rainfall). The gradient aligns with natural perceptions of rain intensity, with darker shades denoting more severe rainfall, which aids in quickly identifying storm-affected areas.

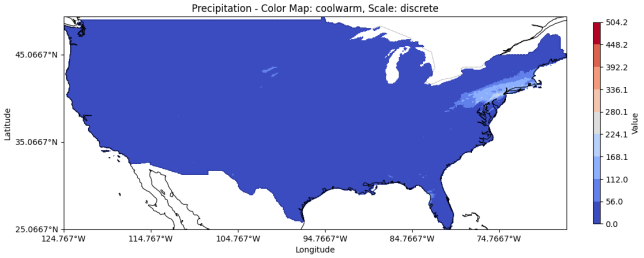


Fig 1.2.1: Map illustrating precipitation levels using the coolwarm (divergent) color palette on a discrete scale.

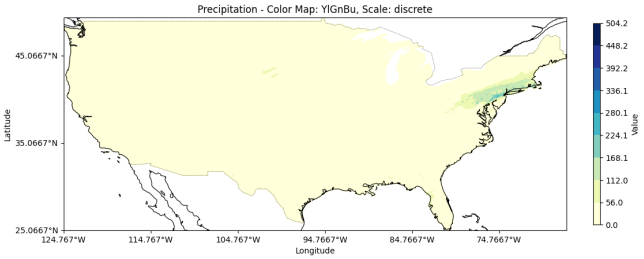


Fig 1.2.2: Map illustrating precipitation levels using the YlGnBu (sequential) color palette on a discrete scale

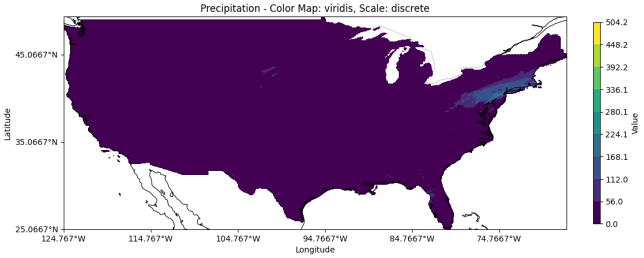


Fig 1.2.3: Map illustrating precipitation levels using the Viridis (sequential) color palette on a discrete scale

Different scales, namely logarithmic, continuous, and discrete, were also tried, and the discrete scaling (Fig. 1.2.2) was found effective for precipitation, as it divides data into specific intervals, each with a distinct color, making it easier to identify ranges, such as low, moderate, and high rainfall. The logarithmic scaling (Fig. 1.2.4) was found to overemphasize lower values, making areas with minimal precipitation appear more intense than they actually are, which can be misleading while making interpretations. Continuous scaling (Fig. 1.2.5) was also useful, though less impactful, in marking distinct rainfall thresholds.

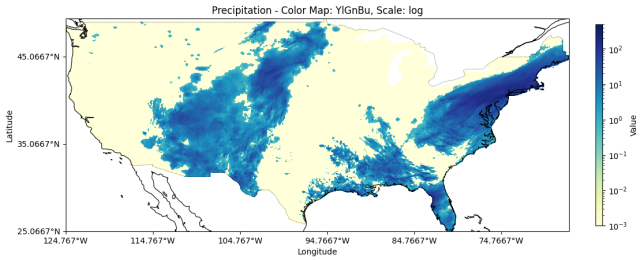


Fig 1.2.4: Precipitation map using YlGnBu palette with logarithmic scale

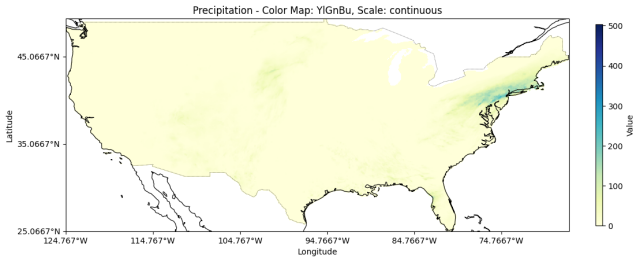


Fig 1.2.5: Precipitation map using YlGnBu palette with continuous scale

b) Temperature: Temperature was tested with the color palettes: hot (moving from yellow to red), coolwarm, and RdYlBu. The hot color palette (Fig. 1.2.7) could not effectively show cooler temperatures. Being a sequential map, it also failed to adequately contrast temperature extremes. The coolwarm palette (Fig. 1.2.6), which is divergent, was found to be ideal for temperature. This palette transitions from blue to red, making it easier to interpret both ends of the temperature spectrum. It provides a balanced view of temperature ranges, with blue hues denoting cooler areas and red hues denoting warmer areas, matching natural temperature perceptions. RdYlBu (Fig. 1.2.8), though a divergent color map is not preferred because it uses contrasting red (to represent lower temperatures) and blue (to represent higher temperatures) colors at opposite ends, creates a misleading perception in temperature data.

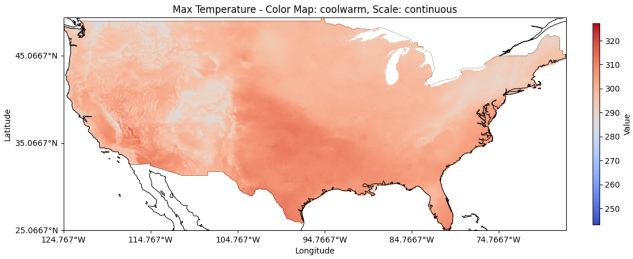


Fig 1.2.6: Map illustrating temperature data using the coolwarm (divergent) color palette on a continuous scale.

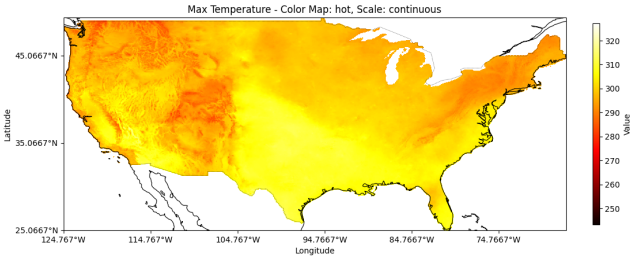


Fig 1.2.7: Map illustrating temperature data using the hot (sequential) color palette on a continuous scale.

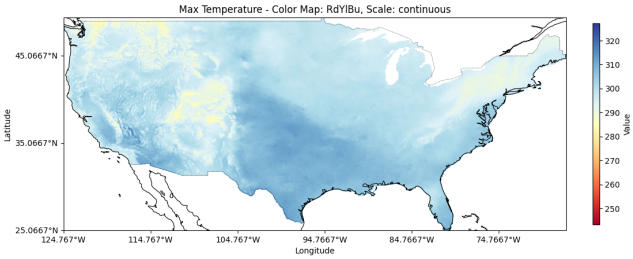


Fig 1.2.8: Map illustrating temperature data using the RdYlBu (divergent) color palette on a continuous scale.

Among the scales, logarithmic scaling (Fig. 1.2.9) was found not to be ideal for temperature as well, as it overemphasizes lower values, making cooler regions appear overly prominent while underrepresenting warmer areas. This is unsuitable for temperature data, where the gradual increase from cool to warm temperatures is more naturally interpreted on a linear or continuous scale. Discrete scaling (Fig. 1.2.10), while useful for marking specific temperature thresholds, introduced unnatural breaks in the temperature gradient, which felt less intuitive. Since temperature changes smoothly over distances, a continuous scale (Fig. 1.2.4) was found to be more preferable than discrete scaling for this variable.

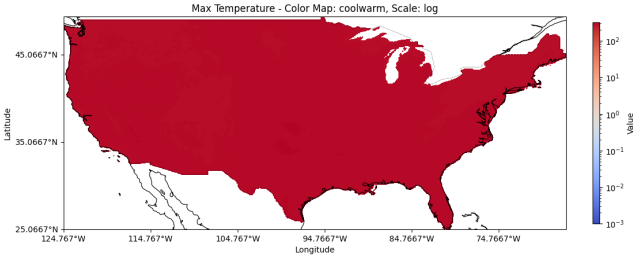


Fig 1.2.9: Temperature map using coolwarm palette with logarithmic scale

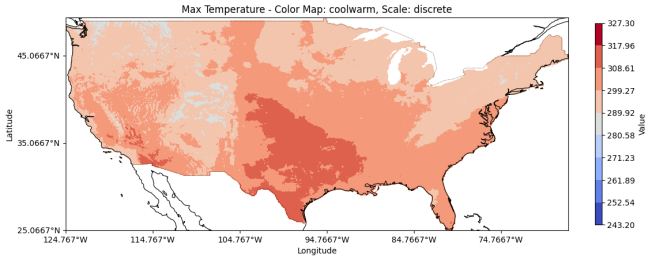


Fig 1.2.10: Temperature map using coolwarm palette with discrete scale

4) Inferences: For temperature data, continuous scaling with the coolwarm color map offered the clearest visualization, and since discrete scaling with the YlGnBu color map was the most effective combination for precipitation data using these two mappings, the colormaps have been plotted for the sampled dates.

a) Precipitation: On August 26, almost the entire U.S. land area shows no rainfall, represented by yellow on the map (indicating 0 rainfall on the YlGnBu color scale) shown in the figure below (Fig. 1.2.11). The only spot of light rain, shown in green, is near Minnesota and South Dakota. This pattern makes sense because Ida was still forming over the Caribbean Sea (as can be seen in Fig. 1) and hadn't yet moved close to the U.S. mainland, so there was no noticeable impact from the storm on land at this stage.

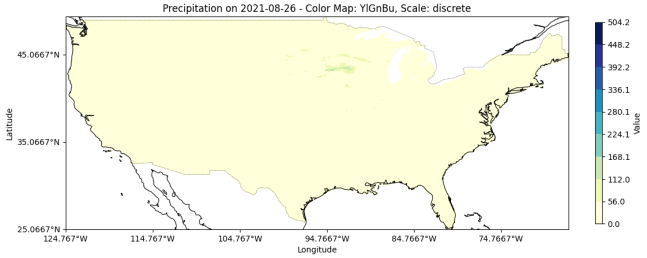


Fig 1.2.11: Color map for 26-08-2021 showing precipitation levels

The maps for August 29 and August 30 show blue-green patches over Louisiana and nearby areas, indicating heavy rain. However, because the scale used for the map has a high global maximum (504 mm), the color isn't dark blue, which would signify extreme rainfall. Instead, it's a green-blue tone, meaning heavy rain compared to other days on this map showing the areas that experienced significant rainfall and flooding when Ida made landfall.

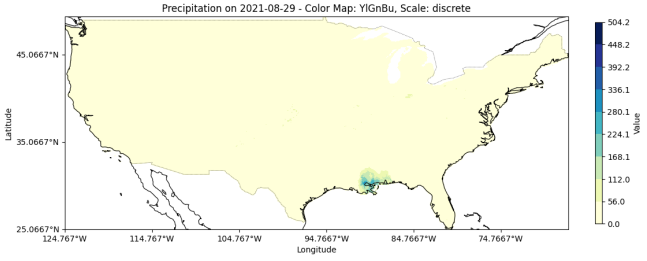


Fig 1.2.12: Color map for 29-08-2021 showing precipitation levels

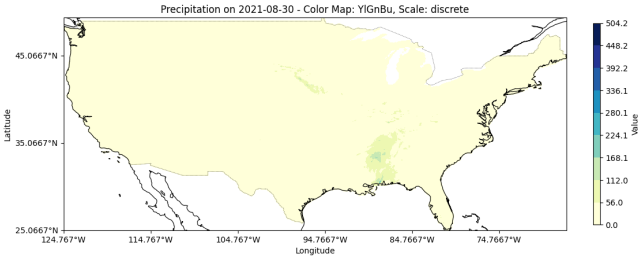


Fig 1.2.13: Color map for 30-08-2021 showing precipitation levels

By September 1, Ida had weakened and moved further inland, bringing a lot of rain to the northeastern U.S., across parts of Pennsylvania, New York, and New Jersey. On the map, these areas show green to blue-green colors, indicating moderate-to-heavy rainfall (but not reaching the maximum). This rainfall was enough to cause serious flash floods and urban flooding, even though it's not shown in the darkest blue due to the global scaling. The map makes it clear where the Northeast experienced impactful rainfall from Ida.

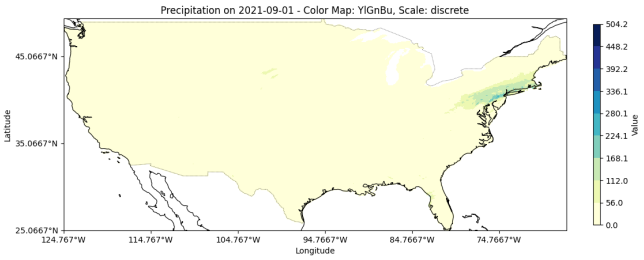


Fig 1.2.14: Color map for 01-09-2021 showing precipitation levels

By September 4, Ida had mostly faded, and there was little rain left, as seen in Fig. 1.2.15. The map shows a mostly yellow landscape (indicating no rain) with a small green spot near Missouri or Kansas, representing light rain. This lack of rain fits with Ida's weakening as it lost its intensity over time.

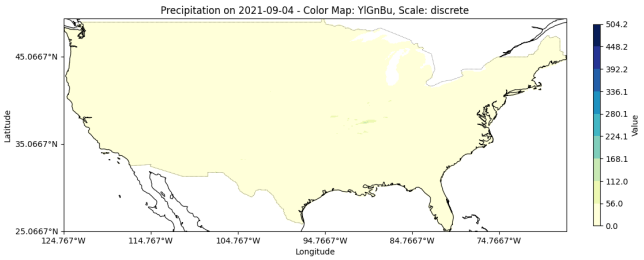


Fig 1.2.15: Color map for 04-09-2021 showing precipitation levels

The precipitation map of September 10 as well shows no rainfall patterns, as the storm had completely dissipated. The color map is uniform, signaling typical weather patterns and marking an end to Ida's influence.

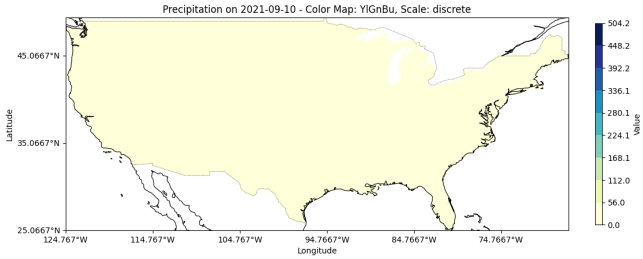


Fig 1.2.16: Color map for 10-09-2021 showing precipitation levels

b) Temperature: On August 26, Ida was forming and strengthening over warm ocean waters in the Caribbean (as can be seen in Fig. 1), but since the map only shows land temperatures, there's no effect visible on land yet. The land areas, especially in the southeastern U.S., show typical warm late-summer temperatures without any influence from the hurricane.

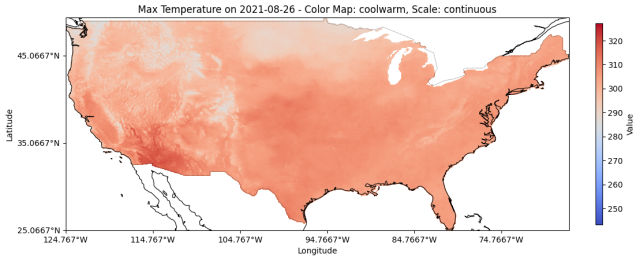


Fig 1.2.17: Color map for 26-08-2021 showing temperature data

When Ida made landfall in Louisiana on August 29, the surrounding area cooled down due to rain and cloud cover from the storm. This cooling effect can be seen on the temperature map around the landfall region, where temperatures drop slightly (represented by a white patch over the area as can be seen in Fig. 1.2.18 and Fig. 1.2.19) compared to the hot, summer-like conditions in other parts of the country.

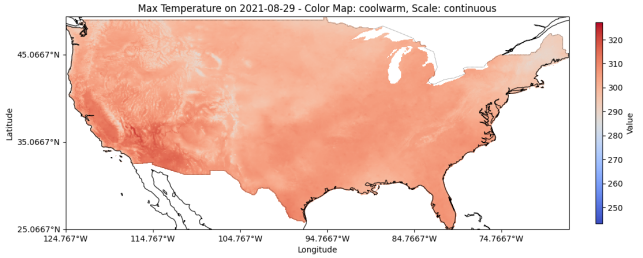


Fig 1.2.18: Color map for 29-08-2021 showing temperature data

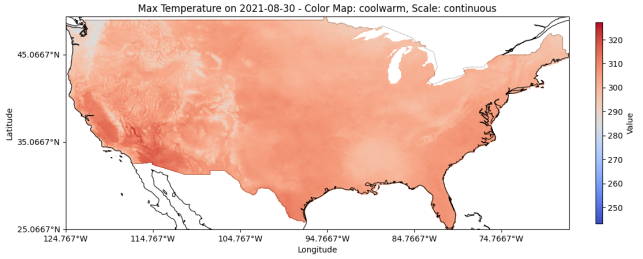


Fig 1.2.19: Color map for 30-08-2021 showing temperature

As Ida traveled inland, it brought cooler temperatures to the Northeast, particularly around New York and New Jersey, where it caused heavy rain and flooding. The storm's clouds and rain made temperatures in the affected areas lower than normal for early September, which is noticeable on the map of September 1.

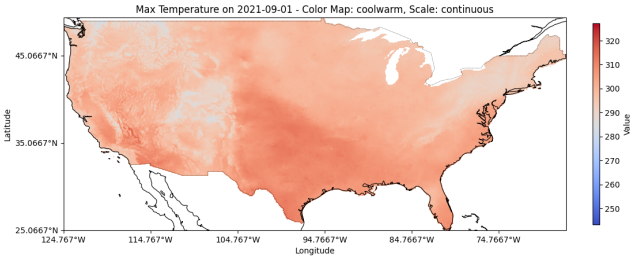


Fig 1.2.20: Color map for 01-09-2021 showing temperature data

By September 4 (Fig 1.2.21), Ida had moved out, but the affected areas were still somewhat cooler, though temperatures were gradually warming up again. By September 10 (Fig. 1.2.22), temperatures in the Northeast and Southeast had mostly returned to their normal seasonal levels, showing that the temporary cooling effect from the hurricane had faded.

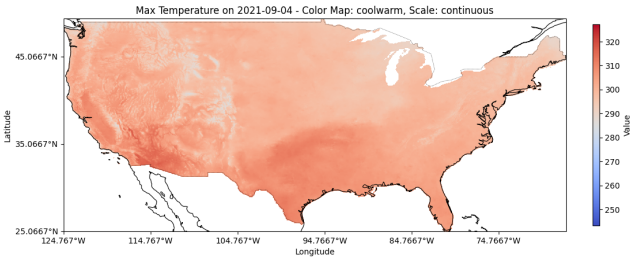


Fig 1.2.21: Color map for 04-09-2021 showing temperature data

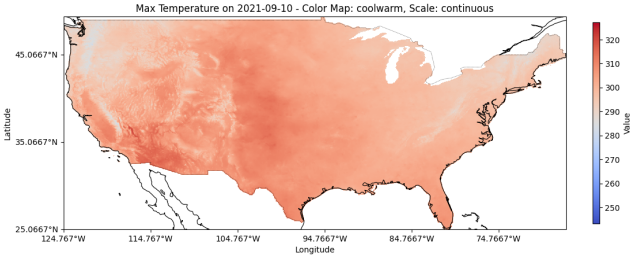


Fig 1.2.22: Color map for 10-09-2021 showing temperature data

C. Contour Mapping

1) Implementation: To visualize vapor pressure deficit and specific humidity data, contour plots were implemented using the Python libraries NetCDF4 [5], Pandas [6], Numpy [14], Cartopy [7], and Matplotlib [8]. The NetCDF4 library NetCDF4 [5] was utilized to read the NetCDF (.nc) files containing meteorological data, while Pandas [6] facilitated date handling and alignment of sample dates between datasets for comparison. NumPy [14] was used to process and manipulate numerical data efficiently. To generate georeferenced contour plots, Cartopy [7] was used for geographic projections and map features like coastlines and borders. Matplotlib [8] was employed to create two types of contour plots: filled contours and line (or marching square) contours. ImageIO [9] was later incorporated to compile the contour plot images into GIFs, allowing for sequential visualization across different sample dates.

2) Experiment: We explored two main visualization methods for representing VPD and SH values: **filled contour plots and marching square contour plots**.

For VPD data, we experimented with the colourmaps, 'YlOrRd' was chosen as it best emphasizes variations in warm tones, making it suitable for representing vapour pressure deficits for the filled contours. For the marching square contours 'viridis' was chosen. For SH data, 'Blues' was chosen for both the contour types, which naturally represents humidity with varying shades of blue. Through experimentation, 15 contour levels were found to provide a balance between detail and visual clarity. Levels below this made it difficult to distinguish finer variations in VPD and SH, while levels above this created visual clutter. Adjustments to line widths and contour boundaries were also tested to improve readability. For instance, the filled contour plot included subtle overlayed contour lines to make transitions between levels clearer.

3) Result:

a) Filled Contour: Referring to Fig 1.3.1 - 1.3.24, The filled contour plot, which uses color gradients to represent changes in vapor pressure deficit (VPD) and specific humidity (SP), provides a smooth, continuous map of Hurricane Ida's intensity and path. This approach effectively captures the hurricane's moisture distribution, displaying how humidity and dryness vary around the storm center. By using color to convey different intensity levels, the filled contour plot highlights the storm's core in a visually impactful way, allowing viewers to instantly identify areas of extreme moisture or low vapour pressure that mark Ida's strongest zones. This continuous representation is particularly useful for storm tracking, as it enables viewers to perceive subtle shifts in humidity and pressure along the path, adding depth and context to the visualization.

b) Marching Square Contour: Referring to Fig 1.3.1 - 1.3.24, the marching square contour plot, which represents data with lines to separate specific value ranges, provides a more segmented view of the hurricane's path. While line contours can be effective for mapping distinct value boundaries, they lack the ability to smoothly display the gradient transitions of moisture and pressure, making the hurricane's trajectory appear more fragmented. The line-based approach limits the representation of the hurricane's intensity

levels, as it only shows where values fall into certain intervals rather than illustrating the full range of changes. This makes it harder to follow Ida's progression and gauge the intensity at various points. Additionally, interpreting line contours typically requires a more analytical approach, as viewers need to assess value boundaries based on line spacing, making it less intuitive than the filled contour plot, especially for a general audience tracking the hurricane's potential impact areas.

4) Inference:

a) Vapour Pressure Depicit (vpd):

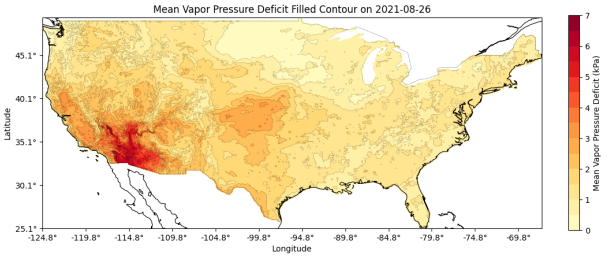


Fig 1.3.1: vpd - filled contour for 26-08-2021

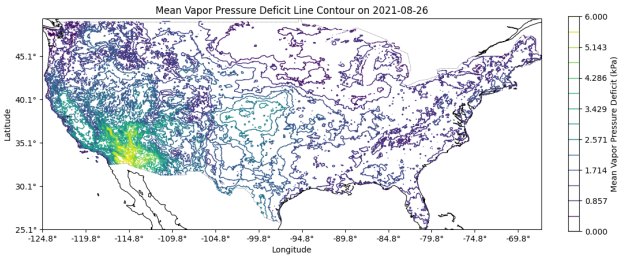


Fig 1.3.2: vpd - marching square contour for 26-08-2021

August 26, 2021: On this date, Hurricane Ida had just formed as a tropical storm in the Caribbean. The contour plot shows that VPD is low in the southeast region of the USA, particularly along the Gulf Coast. This is indicated by the light yellow color, which suggests a high level of moisture in the air, typical for the region during the storm's early formation stages.

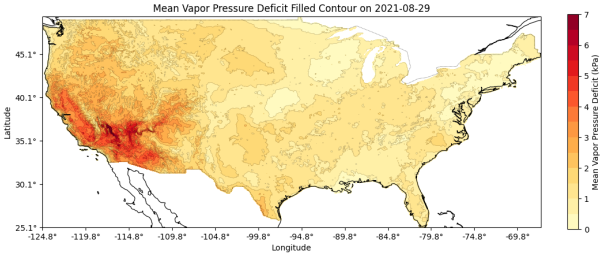


Fig 1.3.3: vpd - filled contour for 29-08-2021

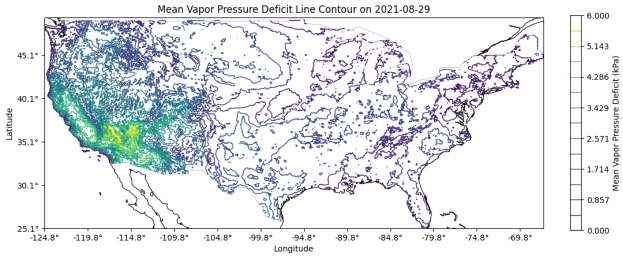


Fig 1.3.4: vpd - marching square contour for 29-08-2021

August 29, 2021: Hurricane Ida made landfall in Louisiana as a Category 4 hurricane. The plot shows that the VPD is the lowest in the area around the hurricane's landfall, marked by the lightest yellow color. This aligns with the hurricane's powerful winds and heavy rainfall, which led to high humidity and low evaporation rates in the region.

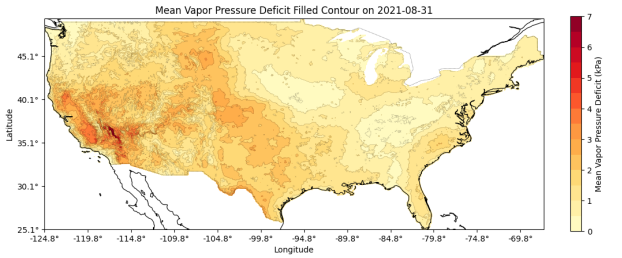


Fig 1.3.5: vpd - filled contour for 31-08-2021

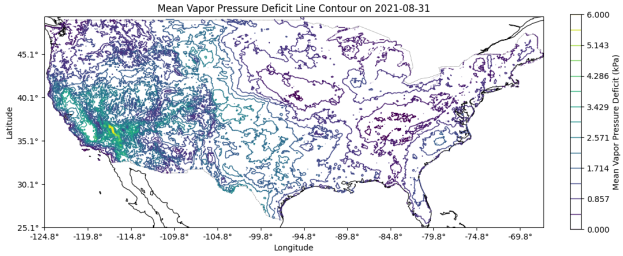


Fig 1.3.6: vpd - marching square contour for 31-08-2021

August 31, 2021: As the hurricane weakened and moved into Mississippi, the contour plot reveals that the VPD remains very low in the path of the storm. Particularly, areas near central Mississippi show low VPD values, indicating that the air in these regions is saturated with moisture due to continued rainfall and cloud cover.

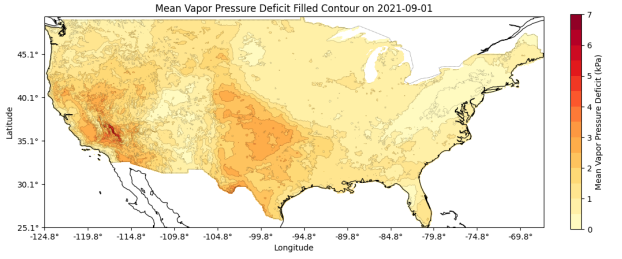


Fig 1.3.7: vpd - filled contour for 1-09-2021

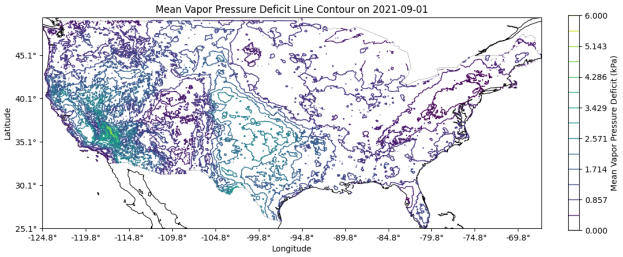


Fig 1.3.8: vpd - marching square contour for 1-09-2021

September 1, 2021: By this time, Ida had weakened further into a tropical depression and moved into the northeastern United States. The VPD in this region, especially around Pennsylvania, New Jersey, and New York, is also shown to be very low. This matches the heavy rainfall and flooding that impacted these areas, as the storm continued to bring moisture-laden air from the Gulf of Mexico.

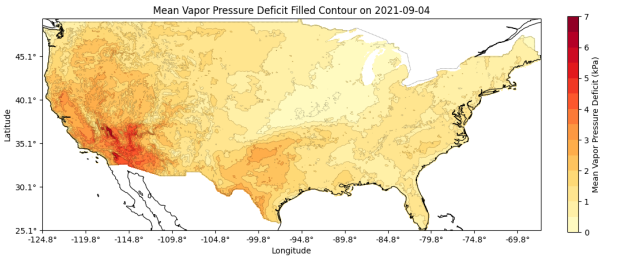


Fig 1.3.9: vpd - filled contour for 4-09-2021

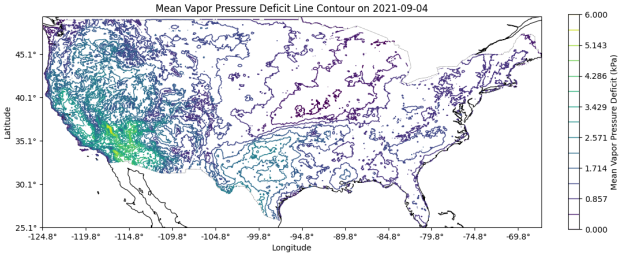


Fig 1.3.10: vpd - marching square contour for 4-09-2021

September 4, 2021: As the storm's remnants dissipated, VPD values gradually returned to more normal levels, though they remained relatively low in the northeastern USA where the storm's moisture effects lingered.

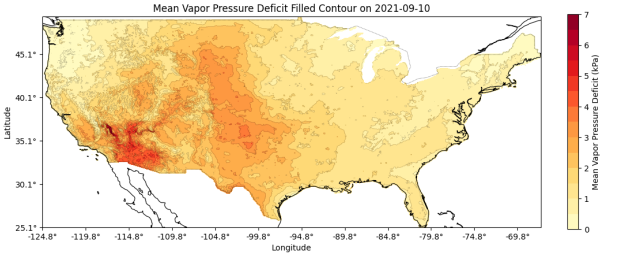


Fig 1.3.11: vpd - filled contour for 10-09-2021

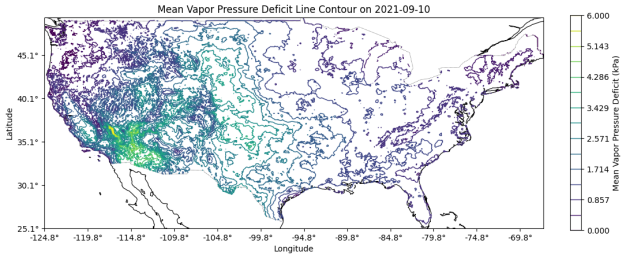


Fig 1.3.12: vpd - marching square contour for 10-09-2021

September 10, 2021: The VPD values in the contour plot continue to show a return to typical levels, with only the southeast still experiencing slightly lower values, which corresponds to the lingering effects of the storm in that region.

Specific Humidity (sh)

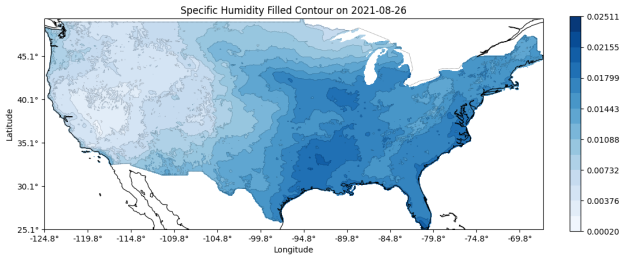


Fig 1.3.12: sh - filled contour for 26-08-2021

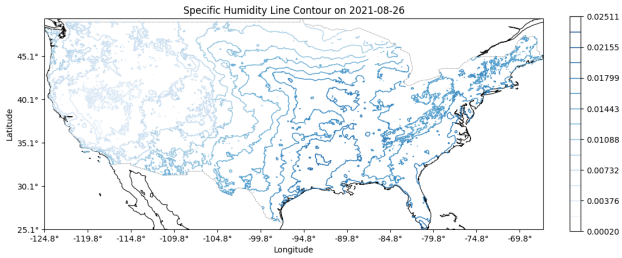


Fig 1.3.13: sh - marching square contour for 26-08-2021

August 26, 2021: The contour plot shows that specific humidity (SH) is highest along the southeast coast, particularly in areas near Florida and the Gulf Coast. This matches the conditions before Hurricane Ida intensified, where the warm, moist air was already present in the region.

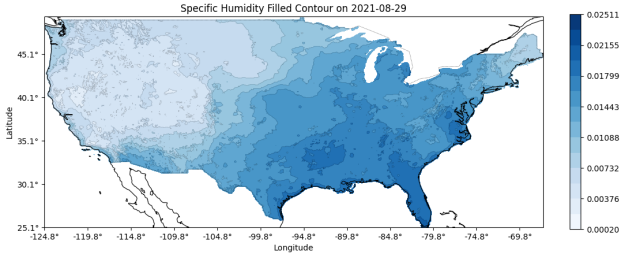


Fig 1.3.15: sh - filled contour for 29-08-2021

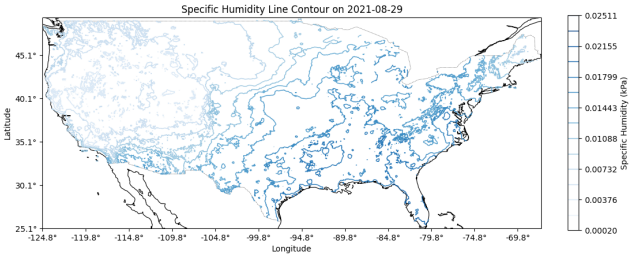


Fig 1.3.16: sh - marching square contour for 29-08-2021

August 29, 2021: As Hurricane Ida made landfall, specific humidity remained high in the southeastern region, as indicated by the deep blue contours. The moisture in the air was extremely high in the areas where the storm was making landfall, consistent with the intense rainfall and winds that were associated with the Category 4 hurricane.

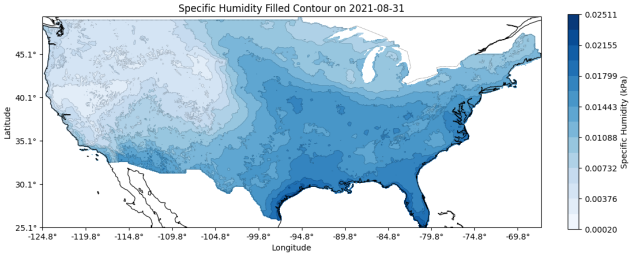


Fig 1.3.17: sh - filled contour for 31-08-2021

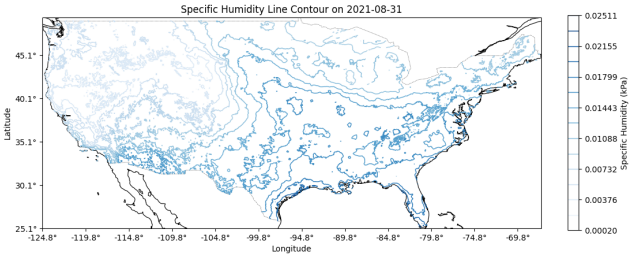


Fig 1.3.18: sh - marching square contour for 31-08-2021

August 31, 2021: By now, Ida had weakened, and the storm's effects were most strongly felt in central Mississippi. The plot shows a decrease in specific humidity in the middle of the country, especially near Nashville. However, the bottom region, particularly the Gulf Coast, still shows high specific humidity values, indicating that the storm was still influencing this area.

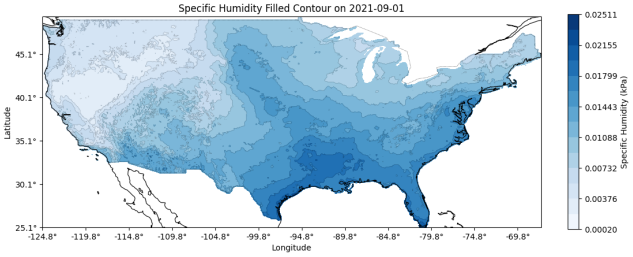


Fig 1.3.19: sh - filled contour for 1-09-2021

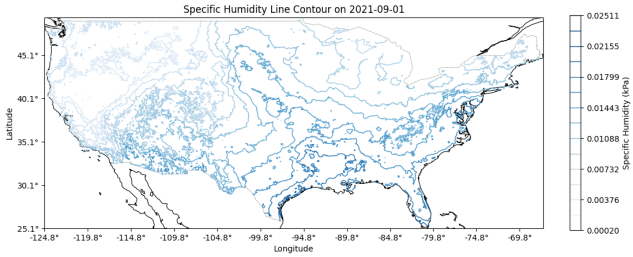


Fig 1.3.20: sh - marching square contour for 1-09-2021

September 1, 2021: Specific humidity further decreases in the middle and northeastern parts of the country, which corresponds to the weakening of the storm as it moved northeast. The regions affected by flash floods in Pennsylvania, New Jersey, and New York saw a significant reduction in specific humidity, which reflects the storm's decreasing intensity.

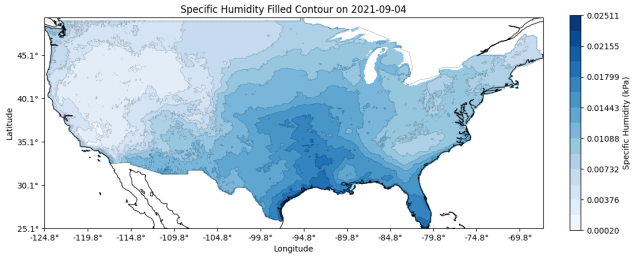


Fig 1.3.21: sh - filled contour for 4-09-2021

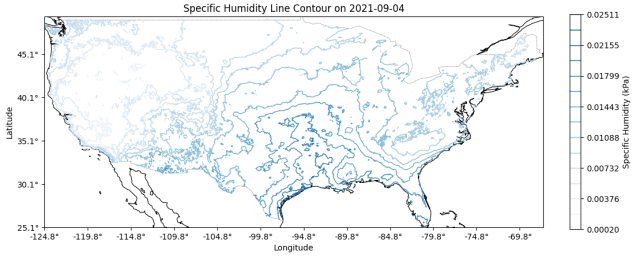


Fig 1.3.22: sh - marching square contour for 4-09-2021

September 4, 2021: As the remnants of Hurricane Ida continued to move away, specific humidity values continued to decrease in the middle and northeast, showing a clear reduction in moisture availability in these regions.

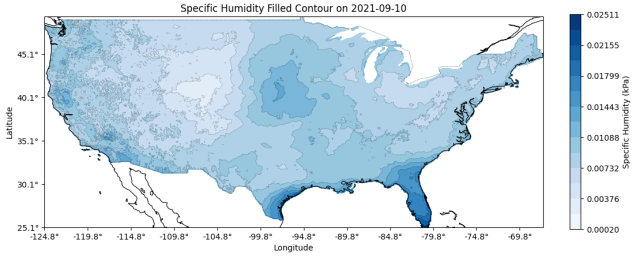


Fig 1.3.23: sh - filled contour for 10-09-2021

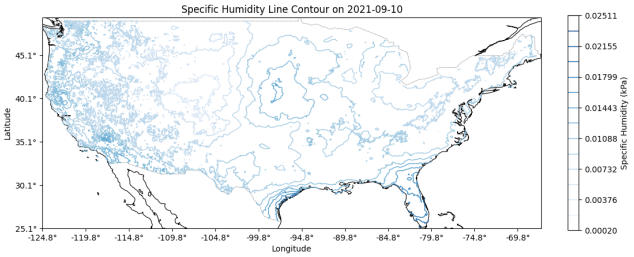


Fig 1.3.24: sh - marching square contour for 10-09-2021

September 10, 2021: With the storm completely dissipated, the southeast coast remained the only region with relatively high specific humidity, while the middle and northeastern regions showed much lower values. This suggests that, while the storm had moved on, it had left behind residual moisture in the southeast, while the rest of the country returned to more typical conditions.

II. Information Visualization

For the node-link diagram, the facebook dataset was used.

The Facebook dataset used is an ego-centric network available from the Stanford Network Analysis Project (SNAP) and can be accessed here. The dataset contains information about 4,039 nodes and 88,234 edges. Each node represents an individual Facebook user in a single ego network, and each undirected edge represents a Facebook friendship between two users. These nodes are anonymous, meaning the dataset does not include identifying information, to protect user privacy.

The dataset includes several files that represent different aspects of the network and user attributes.

- Facebook_combined.txt: contains edges from all the egonets combined.
- nodeId.edges: This file contains the list of edges (friendship connections) in the ego network for the specified nodeId.
- nodeId.circles: This file provides community information, where each community or "circle" is a group of connected users that share certain common attributes or relationships.
- nodeId.feat: This file contains feature data for each user node within the ego network. Each line represents the features for one node, with values indicating specific attributes.
- nodeId.egofeat: Contains the feature vector for the main user (nodeId) around whom the ego network is centered.
- nodeId.featnames: Lists the names of each feature dimension as anonymous identifiers (since real Facebook profile features are not revealed).

For the Parallel Coordinates plot and the Treemap, the Spotify 30000 dataset [11] from Assignment-1 was used.

Dataset info:

- track_id: Song unique ID
- track_name: Song Name
- track_artist: Song Artist
- track_popularity: Song Popularity (0-100), where higher is better
- track_album_id: Album unique ID
- track_album_name: Song album name
- track_album_release_date: Date when album was released
- playlist_name: Playlist name
- playlist_id: Playlist ID
- playlist_genre: Playlist genre
- playlist_subgenre: Playlist subgenre
- danceability: Danceability measure (0.0-1.0)
- energy: Energy measure (0.0-1.0)
- key: Overall key of the track, using Pitch Class notation
- loudness: Loudness of the track in decibels (dB)
- mode: Modality of the track (1 = Major, 0 = Minor)
- speechiness: Measures spoken words in a track
- acousticness: Measures whether a track is acoustic
- instrumentalness: Predicts whether a track contains no vocals
- liveness: Detects the presence of an audience
- valence: Measures the positivity of a track (0.0-1.0)
- tempo: Estimated tempo in beats per minute (BPM)
- duration_ms: Duration of the song in milliseconds

The following data cleanup was done:

- Handling missing data: Songs with any missing values in any of the song features were dropped as they were a low percentage.
- Removal of duplicates: Several songs appeared in multiple playlists. These duplicate entries were dropped to ensure accurate analysis of unique song and artist popularity.

A. Node-Link Diagram

1) Implementation: We utilized networkx [16], a library for managing network graphs, and defined functions to load edge and community (circle) data for each ego network. The main graph, with added ego network IDs, was saved as a GEXF file (facebook_combined_with_ego_network.gexf) for visualization in Gephi [15]. Additionally, individual GEXF files with community data were generated for each ego network (e.g., ego_0_with_communities.gexf) to enable detailed visualization. We used Gephi [15]. to do the following tasks and try different layout algorithms.

a) Visualization of Node Groups Based on Ego Network: The main graph, facebook_combined_with_ego_network.gexf, was loaded into Gephi [15].. Nodes were colored based on their assigned ego network ID, creating a color-coded visualization that highlights each node's origin network and helps identify connections between nodes across ego networks. Intense colour palette was chosen from gephi for visualization as it can show the distinct ego networks clearly.

b) Visualization of Node of Ego_348 Subgraph Based on Community: For a detailed analysis, we focused on the ego_368 subgraph. Nodes in this subgraph were colored based on the circle (community) they belong to within ego_368, using different colors for each community. Intense colour palette was chosen from gephi for visualization as it can show the distinct ego networks clearly.

c) Visualization of Node of Ego_348 Subgraph Based on Degree: We colored the nodes in ego_348 based on their degree (number of connections). Nodes with higher degrees were displayed in darker green color, emphasizing central or highly connected individuals in the network. Green colour is chosen for its better visual appearance

2) Algorithms Used:

a) Force Atlas 2 Algorithm::

- Initialization: Nodes start with an initial, often random, layout.
- Forces: Attractive forces pull connected nodes closer together. Repulsive forces push all nodes apart, creating a balanced distribution of nodes and highlighting communities.
- Scaling and Approximation: Edge weights can be used to adjust the strength of attraction, so stronger connections have more influence on node positions. To improve performance on large graphs, the algorithm uses a Barnes-Hut approximation for faster computation of long-range forces.
- Gravity Control: Additional "gravity" keeps nodes from drifting too far from the center, helping to balance the graph and focus clusters.

b) Yifan Hu Layout::

- Initialization: Nodes are randomly or approximately placed, based on initial values or the desired layout area.
- Forces: Attractive forces bring connected nodes closer together, while repulsive forces push all nodes apart, similar to force-directed layouts.
- Scaling and Approximation: Forces are adjusted to create a well-distributed layout, preventing nodes from clustering too densely in specific areas. Uses a multi-level Barnes-Hut approximation to efficiently calculate forces in dense networks, optimizing performance for large graphs.
- Optimal Distance Control: A target distance between nodes helps keep the layout balanced, ensuring that nodes and communities are spread out without overlaps.

c) Fruchterman-Reingold Algorithm::

- Initialization: Nodes are randomly placed within the drawing area.
- Forces: Attractive forces are applied along edges, pulling connected nodes closer together. Repulsive forces act between all pairs of nodes, pushing unconnected nodes apart to prevent overlap.
- Scaling and Iterations: Forces are scaled to keep nodes evenly distributed across the space and to prevent clustering in one area. A set number of iterations or convergence criteria determines when the algorithm stops.
- Temperature Control: A "cooling" temperature parameter gradually reduces node movement, helping the layout settle into a stable configuration.

d) Radial Axis Layout::

- Initialization: A central "root" node or key node is selected as the center of the layout.
- Layered Placement: Nodes directly connected to the root node are placed in an inner circle, while nodes with indirect connections are arranged in subsequent layers.
- Spacing: Distance from the center is adjusted based on each node's level in the network hierarchy, creating a clear radial structure. Nodes in each layer are spaced evenly around the circumference to prevent overlap and improve readability.
- Degree Influence: Nodes with higher degrees can be placed closer to the center or have larger radii, helping to visually rank nodes by connectivity.

*3) Output - **Visualization of Node Groups Based on Ego Network**:*

a) Force Atlas 2:

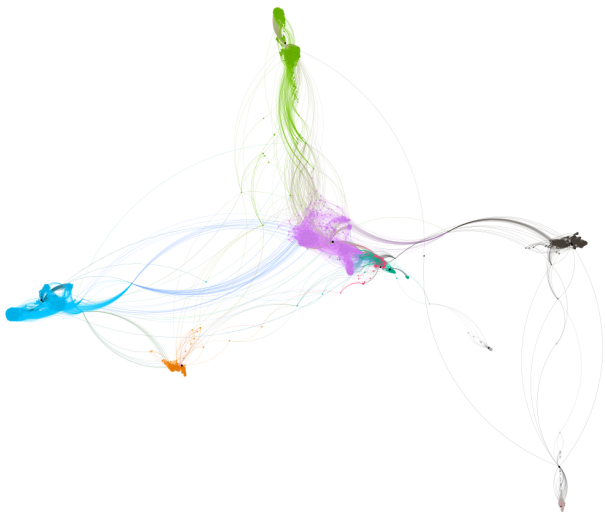


Fig 2.1.1: Force Atlas 2 layout showing ego networks by colour

As shown in fig 2.1.1, the graph effectively separates different ego networks, with 4,039 nodes and 88,234 represented. Each ego network forms distinct clusters of nodes and the nodes are grouped by color, showing clear community structures.

b) Yifan Hu Layout:

Fig 2.1.2: Yifan Hu layout showing ego networks by colour

As shown in fig 2.1.2, the Yifan Hu layout also successfully separates different ego networks, creating a visually balanced graph with nodes distributed in all directions. The nodes are clustered by color, with clear gaps between different ego networks, making it visually appealing and organized.

c) Fruchterman-Reingold: This algorithm took a long time to complete due to the high node count (4,039 nodes) and dense network structure. While it does separate ego networks to some extent, the layout appears crowded, with clusters partially overlapping, making it challenging to distinguish individual groups.

d) Radial Axis Layout:

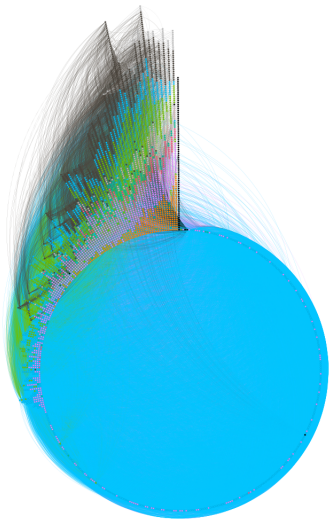


Fig 2.1.3: Radial Axis layout showing ego networks by colour

As shown in fig 2.1.3, the graph is extremely clustered, with nodes stacked tightly in a circular pattern. The high density of edges and nodes results in heavy overlap, creating a solid, filled circle where individual ego networks are indistinguishable. Some nodes, belonging to specific ego networks, peak around one edge of the circle.

Summary - For visualizing all nodes with ego network colors, Force Atlas 2 and Yifan Hu layouts are the most effective for large networks. Force Atlas 2 separated clusters clearly while handling the large node count efficiently, and Yifan Hu achieved a balanced, visually appealing layout with distinct clusters. Fruchterman-Reingold struggled with long processing time and crowding due to the dense network, while the Radial Axis Layout is unsuitable as it created excessive overlap, rendering clusters indistinct.

*4) Output - **Visualizing the Ego_348 Subgraph with Community Colors**:*

a) Force Atlas 2:

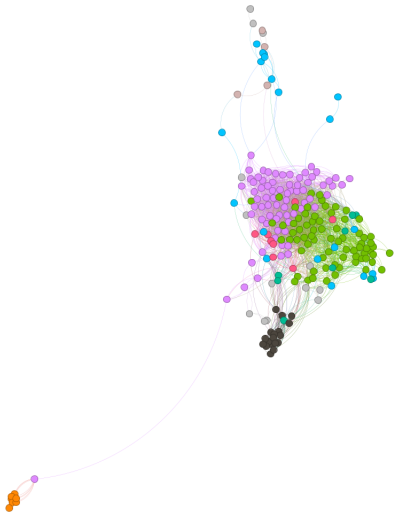


Fig 2.1.4: Force Atlas 2 showing communities in ego_348 subgraph by colour

As shown in fig 2.1.4, the graph effectively separates nodes belonging to different communities, grouping nodes of the same color together. Communities with a higher number of nodes are more centralized, while smaller communities are positioned further from the center. This layout allows for clear visualization of distinct community clusters.

b) Yifan Hu:

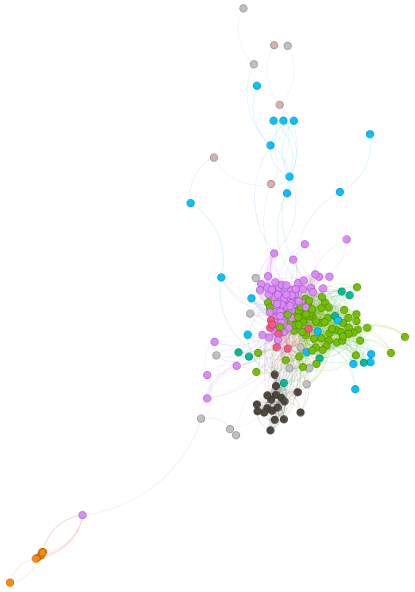


Fig 2.1.5: Yifan Hu showing communities in ego_348 subgraph by colour

As shown in fig 2.1.5, the graph effectively separates nodes belonging to different communities, with nodes in the same color forming tight clusters. This layout is well-spread, with high-node communities near the center and smaller communities more dispersed, making community divisions easily distinguishable and visually balanced.

c) Fruchterman-Reingold:

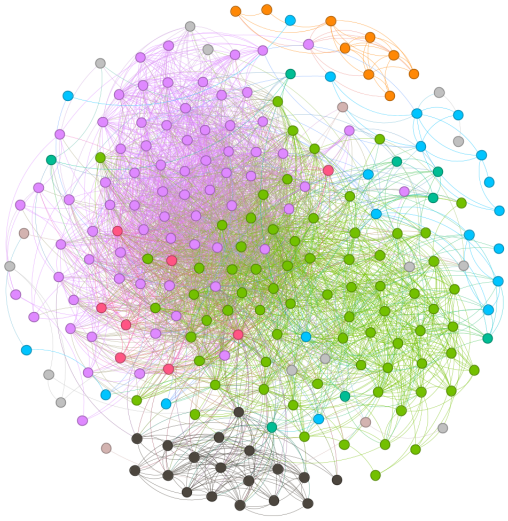


Fig 2.1.6: Fruchterman-Reingold showing communities in ego_348 subgraph by colour

As shown in fig 2.1.6, this layout also distinguishes nodes by community, clustering nodes of the same color together. Despite this, large communities are still somewhat centralized, while smaller ones are closer to the periphery. There are no defined boundries seen for the communites.

d) Radial Layout:

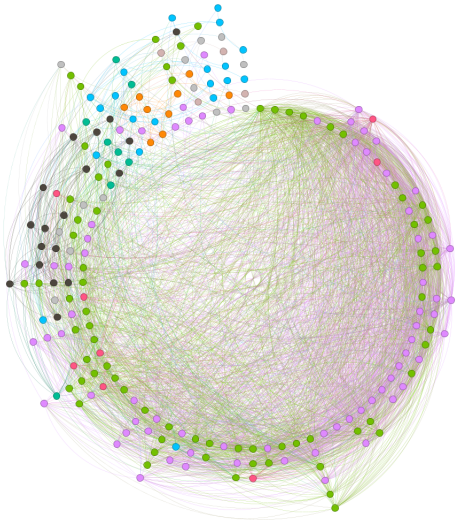


Fig 2.1.7: Radial Layout showing communities in ego_348 subgraph by colour

As shown in fig 2.1.7, the graph is highly clustered, making it difficult to distinguish between communities. Nodes are arranged in layers radiating outward, causing overlap within each community. Large communities dominate the central area, while smaller communities are pushed to the outer edges, resulting in a dense, circular pattern with heavy edge overlap, making community separation nearly impossible to discern. Summary - For visualizing the Ego_368 subgraph with community colors, Force Atlas 2 and Yifan Hu provide the clearest representations, with distinct, well-separated community clusters. Fruchterman-Reingold is moderately effective but suffers from some crowding, while the Radial Layout is unsuitable due to excessive clustering and overlapping, which obscure community distinctions. Force Atlas 2 and Yifan Hu are the most ideal choices for this task.

5) Output - ***Visualization of Node of Ego_348 Subgraph Based on Degree****:*

a) Force Atlas 2:

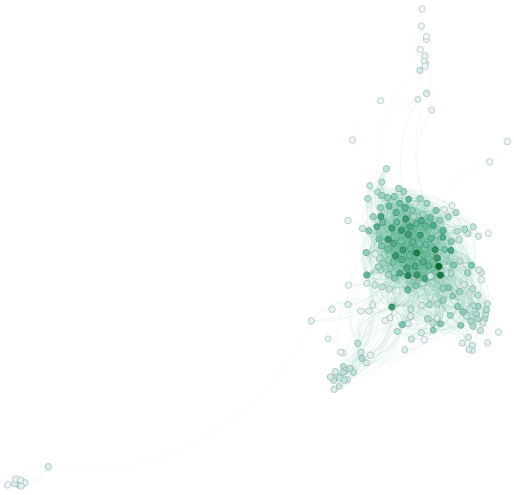


Fig 2.1.8: Force Atlas 2 showing high degree nodes in ego_348 subgraph

As shown in fig 2.1.8, in this layout, nodes with a high degree (representing influential figures in the Facebook network, such as those with many connections) are centralized, while nodes with fewer connections are distributed toward the outer edges. The high-degree nodes being close together at the center highlights their interconnected nature, suggesting they may represent key members who bridge different parts of the network. However, due to the proximity of nodes, individual details can be harder to distinguish.

b) Yifan Hu:

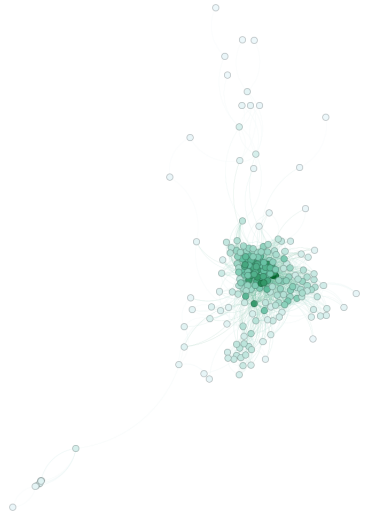


Fig 2.1.9: Yifan Hu showing high degree nodes in ego_348 subgraph

As shown in fig 2.1.9, Similar to Force Atlas, high-degree (influential) nodes appear in the center, with lower-degree nodes extending outward. However, the Yifan Hu layout offers a bit more separation, allowing the connections between influential figures and surrounding nodes to be more visible. This layout effectively highlights the central hub formed by highly connected individuals and provides a balanced view of connections radiating outward, making it easier to identify the network's structure and influence patterns.

c) Fruchterman-Reingold:

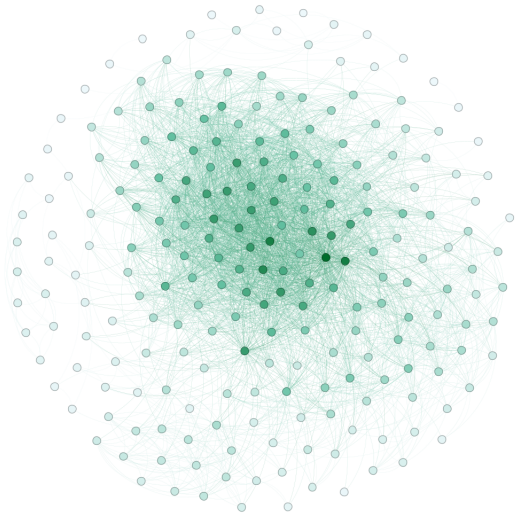


Fig 2.1.10: Fruchterman-Reingold showing high degree nodes in ego_348 subgraph

As shown in fig 2.1.10, in this layout, the high-degree nodes are at the core, surrounded by a nearly circular arrangement of lower-degree nodes. This clear separation makes the influential figures particularly noticeable, standing out as a central cluster. The high-degree nodes' central position surrounded by others with fewer connections illustrates a hierarchical structure, where key individuals act as connectors within the community, making them easily identifiable as primary network hubs.

d) Radial Layout:

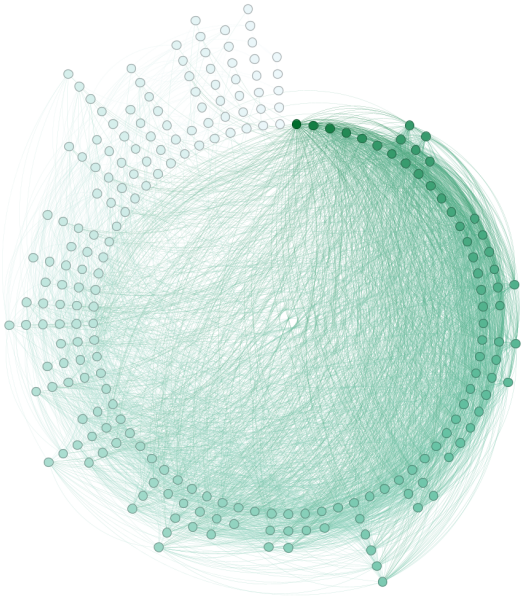


Fig 2.1.11: Radial Layout showing high degree nodes in ego_348 subgraph

As shown in fig 2.1.11, Here, nodes are arranged in a clockwise fashion according to degree, with high-degree nodes closer to the center and lower-degree nodes radiating outward. While this setup theoretically separates nodes by connection count, stacking among nodes, especially those with fewer connections, can make it difficult to view individual relationships clearly. This layout provides a less intuitive sense of the network structure and influence flow, as it is harder to see the interactions among central and peripheral nodes due to overlap.

Summary - For visualizing the Ego_368 subgraph with degree-based colors, Fruchterman-Reingold and Yifan Hu offer the clearest insights into the Facebook network's hierarchical structure. Both layouts highlight the central placement of high-degree (influential) nodes, making it easy to see which individuals are most connected and likely serve as key figures in the network. Force Atlas 2 also offers a good representation but with slightly denser clustering, while the Radial Layout is less ideal due to node stacking, which obscures the relationship between central and peripheral nodes.

B. Parallel Coordinates Plot

1) Implementation: The parallel coordinates plot for visualizing Spotify song features was implemented using a combination of JavaScript and multiple libraries. The preprocessing and formatting of data into a CSV file were carried out using Pandas [6]. D3.js [12] was utilized for loading and processing the CSV data directly in the browser, while Plotly.js [13] with its `parcoords` package was used to create the parallel coordinates plot.

Each axis in the plot represents a different feature from the dataset. The axes include: 'track_popularity', 'genre_id', 'danceability', 'energy', 'key', 'loudness', 'mode', 'speechiness', 'acousticness', 'instrumentalness', 'liveness', 'valence', 'tempo', and 'duration_ms'. Each line represents a song, and the lines are color-coded by genre (pop, rap, rock, latin, r&b, edm) using a discrete jet colormap.

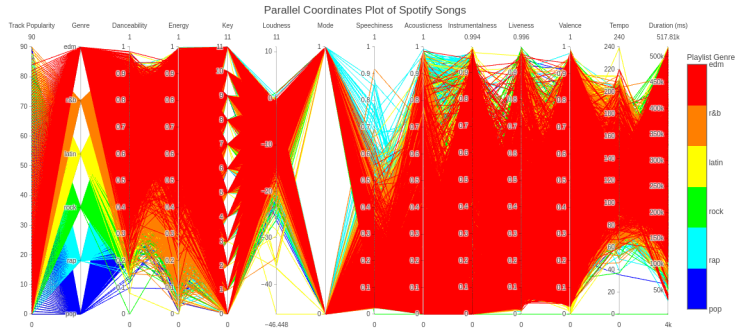


Fig 2.2.1: Parallel coordinates plot of Spotify song features

2) User interactions: For user interactions, the following features were implemented:

- **Brushing** - Users can click and drag on any axis to filter the range of values shown for that attribute. For instance, brushing along the track popularity axis highlights the most popular songs, dimming the rest of the plot for better visibility. Brushing can be applied across multiple axes simultaneously, as seen in Fig 2.2.2.
- **Axis Reordering** - Users can drag and rearrange axes to bring attributes of interest closer together, facilitating direct comparison of different features. For example, Fig 2.2.2 demonstrates energy and loudness being brought adjacent to genre for better comparison.

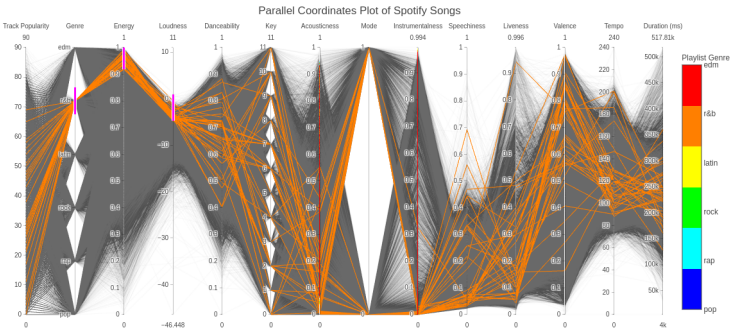


Fig 2.2.2: Parallel coordinates plot of Spotify song features with brushing applied on genre (r&b), energy, and loudness axes. Axes reordered to position energy and loudness closer to genre.

3) Inferences:

- **Top Songs Across Genres**
  As shown in Fig 2.2.3, brushing was applied on the track popularity axis to select the most popular songs.
  1) The majority of the most popular songs belong to the pop and latin genres.
  2) Highly popular songs are few in number in the rap, rock, r&b, and edm genres.
  3) Top songs tend to have low values of instrumentalness, liveness, and duration.
  4) Valence (musical positivity) is not particularly high for the top songs.

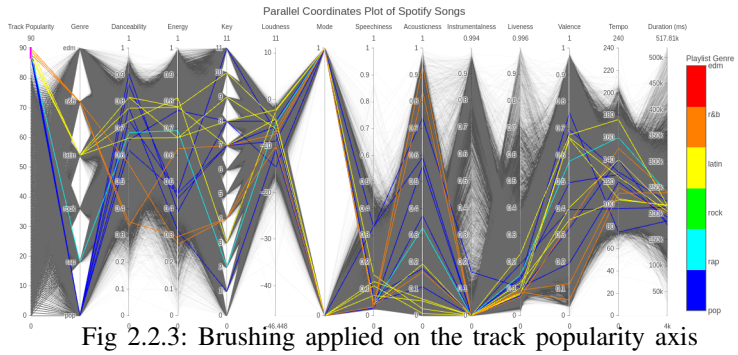


Fig 2.2.3: Brushing applied on the track popularity axis to select the most popular songs.

- **Energy vs Danceability**
  As shown in Fig 2.2.4, brushing was applied on the energy axis to select high-energy songs.
  1) There is no clear proportionality between energy and danceability. High-energy songs cover the entire range of danceability.
  2) EDM songs mostly fall in the high-energy spectrum, with very few low-energy tracks. High-energy EDM songs exhibit low acousticness and medium tempo values.

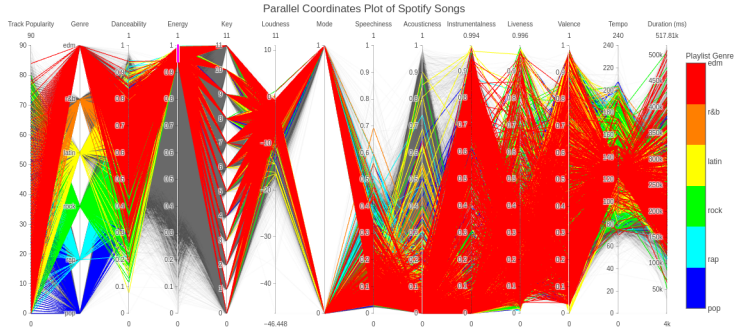


Fig 2.2.4: Brushing applied on the energy axis to select high-energy songs.

- **Speechiness**
  As shown Fig 2.2.5, brushing was applied on the speechiness axis to select songs with high speechiness.

  Songs with the highest speechiness values are predominantly rap songs, consistent with the genre's lyrical emphasis.

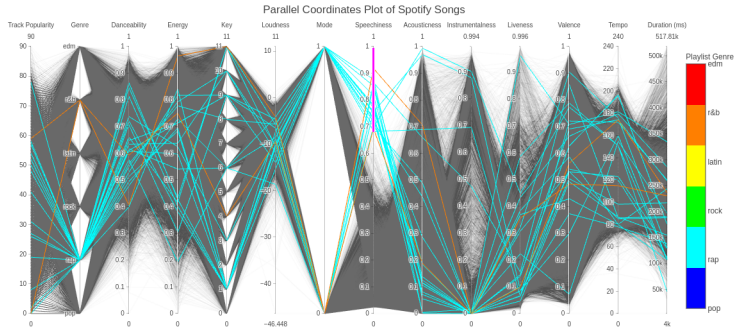


Fig 2.2.5: Brushing applied on the speechiness axis to select songs with high speechiness.

- **Loudness**
  As shown in Fig 2.2.6, brushing was applied on the loudness axis to select the quietest songs.
  Songs with the lowest loudness values are primarily from the latin genre.

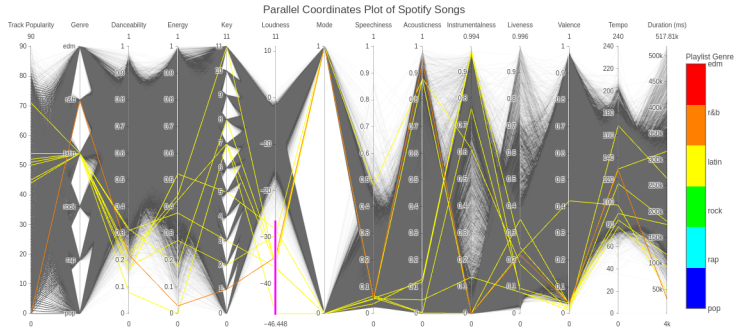


Fig 2.2.6: Brushing applied on the loudness axis to select the quietest songs.

- **Energy vs Loudness**
  As shown in Fig 2.2.7, brushing was applied on the loudness axis to select the loudest songs. The axis order was adjusted to place loudness next to energy.

  A strong positive correlation is observed between energy and loudness; louder tracks tend to have higher energy.

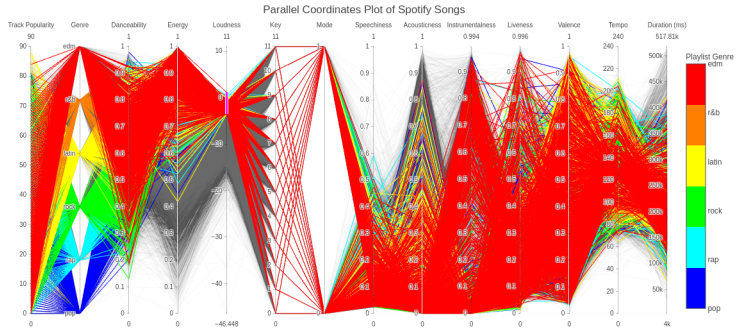


Fig 2.2.7: Brushing applied on the loudness axis to select the loudest songs. The axis order was adjusted to place

loudness next to energy.

- **Energy vs Acousticness**
  As shown in Fig 2.2.8, brushing was applied on the energy axis to select high-energy songs. The axis order was adjusted to bring energy and acousticness closer together.
  Energy and acousticness are inversely proportional; high-energy songs exhibit lower acousticness.

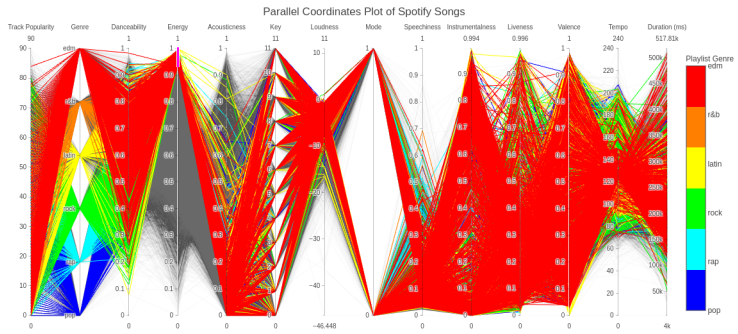


Fig 2.2.8: Brushing applied on the energy axis to select high-energy songs. The axis order was adjusted to bring energy and acousticness closer together.

C. Treemap

1) Implementation: Interactive treemap visualizations were implemented using JavaScript, HTML, Plotly [13], and D3.js [12], showcasing different aspects of the Spotify song dataset [11]. Three treemaps were experimented with, representing various views of the dataset, each built upon preprocessed data and organized through unique hierarchical structures to reflect the popularity of genres, subgenres, and artists. User interactions, including hover and click functionalities, were added to enhance data exploration.

The initial data cleanup of removing the songs with missing values was followed by deduplication of tracks within key groupings to ensure accurate results for each visualization. The data was segmented into three distinct CSV files. The first file (genre_subgenre_data.csv) contains average popularity for each genre and subgenre, ranked by popularity within genres. The second file (year_artist_data.csv) includes the top 10 artists per year, based on average popularity, with an additional column for yearly average popularity to aid in color scaling. The third file (genre_artist_popularity_trackcount.csv) lists top artists by genre, showing each artist's popularity and unique track count within their genre. These structured CSV files allowed for efficient loading and rendering in JavaScript for each treemap experiment.

2) Partioning strategies for Treemap Layout: To display the hierarchical data effectively, four partitioning strategies have been implemented: Squarify, Slice, Dice, and Slice-Dice. These methods affect the distribution and organization of data blocks within the treemaps. The Squarify layout (Fig. 2.3.1) is designed to create visually balanced rectangles, making it easier to identify patterns across categories. The slice layout (Fig. 2.3.3) divides the space in rows, while the dice layout (Fig. 2.3.4) splits it into columns, and slice-dice (Fig. 2.3.2) alternates between rows and columns, allowing for a clean hierarchical view.

Fig 2.3.1: Treemap with Squarify layout for Top Artists by Genre and Track Count

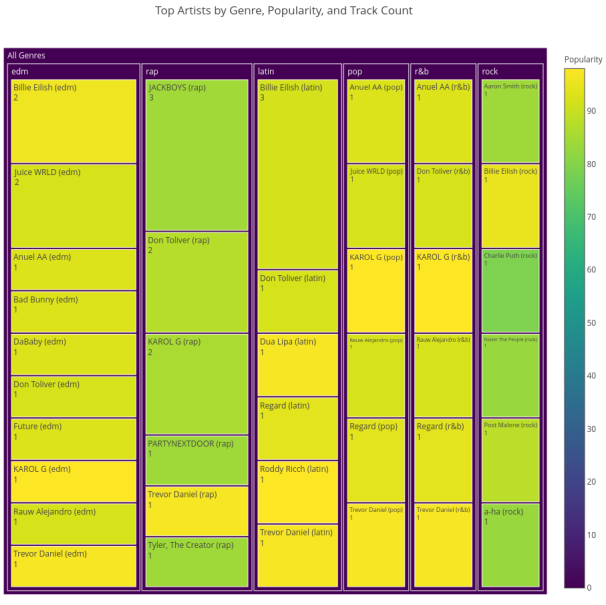


Fig 2.3.2: Treemap with Slice-Dice layout for Top Artists by Genre and Track Count

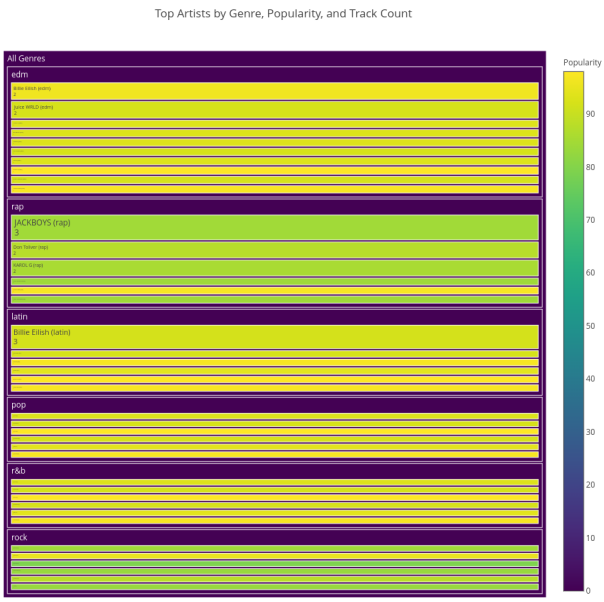


Fig 2.3.3: Treemap with Slice layout for Top Artists by Genre and Track Count

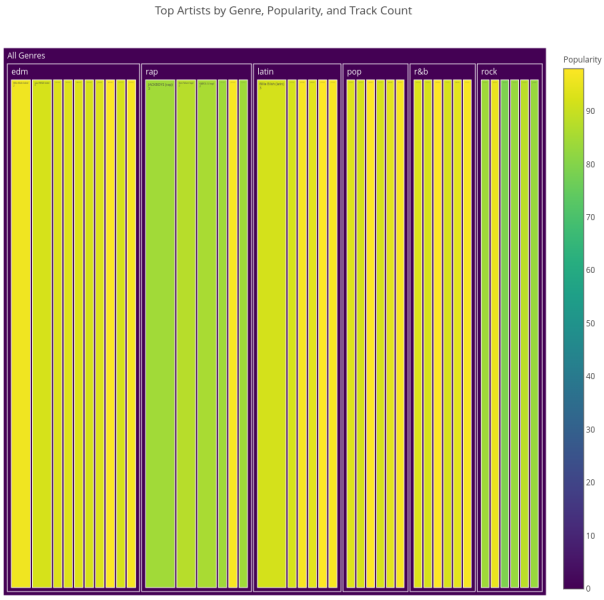


Fig 2.3.4: Treemap with Dice layout for Top Artists by Genre and Track Count

3) User Interactions and interactivity features: Plotly's treemap interactions help in increasing usability and interactivity. Blocks within the treemaps can be clicked to zoom in on subcategories, allowing a focused view of a particular genre or artist set. This zooming feature, a standard option in Plotly, is ideal for examining specific data points within larger categories.

Fig 2.3.5: A closer examination of the subgenre rap, obtained by clicking on it from the treemap of Top Artists by Genre and track count - user interaction.

Also in each treemap, the hover interactions are powered by Plotly's built-in hover feature, which displays additional details dynamically. On hovering over any block in the treemap, instantly details such as the artist's or subgenre's popularity, rank, and track count, etc. can be seen. This feature provides specific insights without requiring additional clicks or navigating away, allowing for a seamless and informative exploration experience. In Plotly, these hover details can be customized to include relevant metadata from the dataset.

4) Inferences: Three different treemaps were plotted, each providing a unique insight into the dataset, with colors and sizes representing different metrics. A dropdown with color palette choices is implemented in the browser to allow the user to try out and select their desired palette, and of them, the Viridis color map was found to best represent all three treemaps. This choice was motivated by its smooth gradient and high contrast, which makes it easier to distinguish different popularity levels. Viridis not only enhances readability but also maintains a consistent look for all people, including those with color vision deficiencies.

a) Top Artists by Genre and Track Count: The slice-dice layout was found to best capture hierarchical relationships in this treemap, as can be seen in Fig. 2.3.2. Each genre is allocated a proportional area, which is then divided among its artists based on track count. This layout aligns well with the genre-based hierarchy, preserving readability and making artist-specific contributions within each genre clear.
Here, block size represents an artist's track count within a genre, offering insight into the artist's contribution or

output level within that genre. The color gradient represents popularity and helps distinguish prolific artists who also have high popularity from those with a high track count but lower popularity.

This treemap sheds light on productivity versus popularity trends, revealing that some artists may have a smaller but intensely popular set of tracks (eg., Trevor Daniel, Dua Lipa), while some others maintain high track output but with a moderate popularity level (eg., Jackboys). These differences offer insight into how productivity and popularity interact within genres; this also helps in identifying influential artists within each genre and comparing the distribution of popularity among artists with varying levels of productivity.

b) Genre and Subgenre Popularity Treemap: For this treemap, both Slice-Dice (Fig. 2.3.6) and Squarify (Fig. 2.3.7) layouts were found to be performing well. In the slice-dice layout, genres are arranged in columns based on popularity, with the most popular subgenres appearing at the top of each genre's column, making it easy to visually grasp leading subgenres within each main genre. But in this case, since the number of subgenres in a genre is less, it is easy to identify the popular subgenres even in the squarified layout. But in cases where there would be more number of sub-blocks, then using slice-dice would have more benefit. The size of each block represents the popularity of each subgenre relative to its parent genre. This setup enables users to see how each genre is composed of various subgenres and which subgenres are driving the genre's overall popularity. Blocks that are larger and more in shades of yellow represent the more popular subgenres within a genre, while smaller and greenish-blue blocks show those that are less popular.

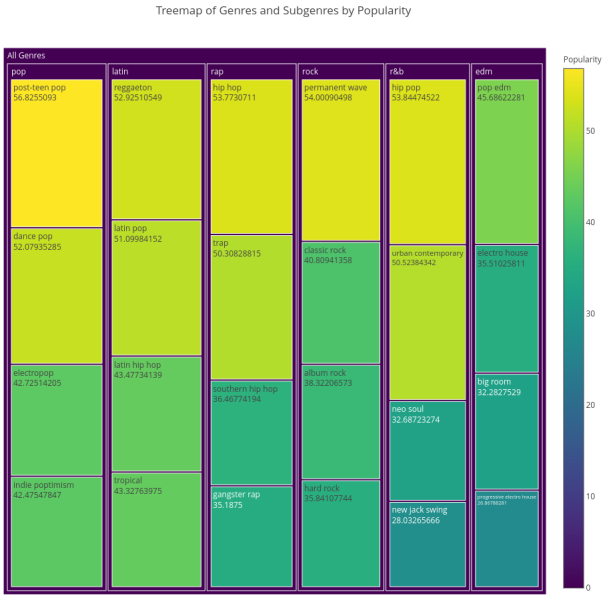


Fig 2.3.6: Popularity Distribution of Subgenres within Genres, with size representing subgenre popularity. (slice -dice layout)

Fig 2.3.7: Popularity Distribution of Subgenres within Genres, with size representing subgenre popularity. (squarify layout)

Fig 2.3.8: Visualization of the Top 10 Artists by Year, revealing year-over-year shifts in artist rankings and highlighting the most prominent artists per year.

c) Top 10 Artists by Year Treemap: For this treemap, the Squarify layout was chosen to showcase the most popular artists in each selected year (Fig. 2.3.8). Squarify ensures balanced rectangles, making moderately popular artists more visible while preventing high-popularity artists from dominating the view. Year range selection in the browser allows users to dynamically filter the data by start and end

year.

This treemap offers a quick visual snapshot of the most popular artists each year, making it easy to identify which artists gained or sustained prominence over time. Each artist's block is color-coded using the Viridis color map, where more yellow shades represent higher popularity. The year blocks themselves are also shaded according to the cumulative popularity of the top artists within that year, providing a high-level view of each year's music trends.

This visualization helps in observing recurring appearances of certain artists across multiple years, signaling lasting popularity, or in detecting new artists emerging in specific years due to some major releases.

III. Authors' contributions

Data preparation and cleanup was done by all the team members.

- Saniya Ismail Kondkar: Contour mapping, Node-link diagram
- Ragini Metlapalli: Colour mapping, Treemap
- Dyuthi Vivek: Quiver plot, Parallel coordinates plot

References

[1] Wikipedia. (2021). *Hurricane Ida*. Retrieved from https://en.wikipedia.org/wiki/Hurricane_Ida

[2] Weather.com. (2021). *Hurricane Ida recap: Louisiana, South, Northeast*. Retrieved from https://weather.com/storms/hurricane/news/2021-09-02-hurricane-ida-recap-louisiana-south-northeast

[3] gridMET dataset https://www.climatologylab.org/gridmet.html

[4] Xarray documentation https://docs.xarray.dev/en/stable/

[5] NetCDF4 documentation https://unidata.github.io/netcdf4-python/

[6] Pandas documentation https://pandas.pydata.org/docs/

[7] Cartopy documentation https://pypi.org/project/Cartopy/

[8] Matplotlib documentation https://matplotlib.org/

[9] Imageio documentation. Retrieved from https://imageio.readthedocs.io/en/stable/

[10] SciPy documentation https://scipy.org/

[11] Spotify 30000 dataset https://www.kaggle.com/datasets/joebeachcapital/30000-spotify-songs?select=spotify_songs.csv

[12] D3.js documentationhttps://d3js.org/getting-started

[13] Plotly.js documentation https://plotly.com/javascript/

[14] NumPy documentation. https://numpy.org/doc/stable/

[15] Gephi documentation. https://gephi.org/users/

[16] NetworkX documentation. https://networkx.github.io/documentation/stable/