# Expanding and Strengthening the Transition from NCL to Python Visualizations

Jiaqi Li[1], Erin Lincoln[2], Michaela Sizemore[3], Anissa Zacharias[3], Orhan Eroglu[3], Julia Kent[3]

[1] Middlebury College, Middlebury, VT, jiaqil@middlebury.edu
[2] Brown University, Providence, RI, erin_lincoln@brown.edu
[3] National Center for Atmospheric Research, Boulder, CO

## Background

- In January of 2019, NCAR announced the transition from NCAR Command Language (NCL) to the Python scientific ecosystem.
- NCL is put into maintenance mode. Major bugs of the language would be fixed but new features are not being developed.
- Python supports a robust scientific environment for computation and visualization with its related packages.
- GeoCAT team is in charge of making the transition as smooth as possible.
- There are three main components to GeoCAT efforts.
  - GeoCAT-Comp
  - GeoCAT-Viz
  - GeoCAT-Examples
- GeoCAT-Comp makes the transition from NCL computational routines into pure python.
- GeoCAT-Viz and GeoCAT-Examples support the transition of scientific visualizations from NCL to Python ecosystem.
- SIParCS project Expanding the GeoCAT-Examples Visualization Gallery was established in summer 2020 and continued in summer 2021

GeoCAT-Examples Gallery

GeoCAT-Examples GitHub

GeoCAT-Viz GitHub

## Why Convert from NCL to Python?

- Python is widely used by the scientific community, allowing for community input
- Packages like Matplotlib and Cartopy can replicate NCL functionalities almost exactly and allow for even more use cases and functionality
- Python is relatively easy to understand, even for novice programmers

## GeoCAT Aiding the Transition

### GeoCAT-Examples

*A Collection of Python Scripts for Recreated Plots*

- Contains a diverse set of NCL plots recreated in Python scientific ecosystem
- Includes the complete Python script and documentations of each plotting component
- Explains differences between the NCL and recreated plot
- Serves as plotting templates that recreate NCL counterparts almost exactly
- Serve as resources and teaching tools for scientific visualizations

**135+ Recreated Plots**

**20+ Plotting Categories**

### GeoCAT-Viz

*A Collection of Functions to Create NCL-Like Visualizations in Python*

- Contains a library of utility functions
- Wraps around repeated codes for common plotting components
- Wraps around computational/plotting routines
- Called by NCL Plot (see right bottom) to generate higher-level plotting templates

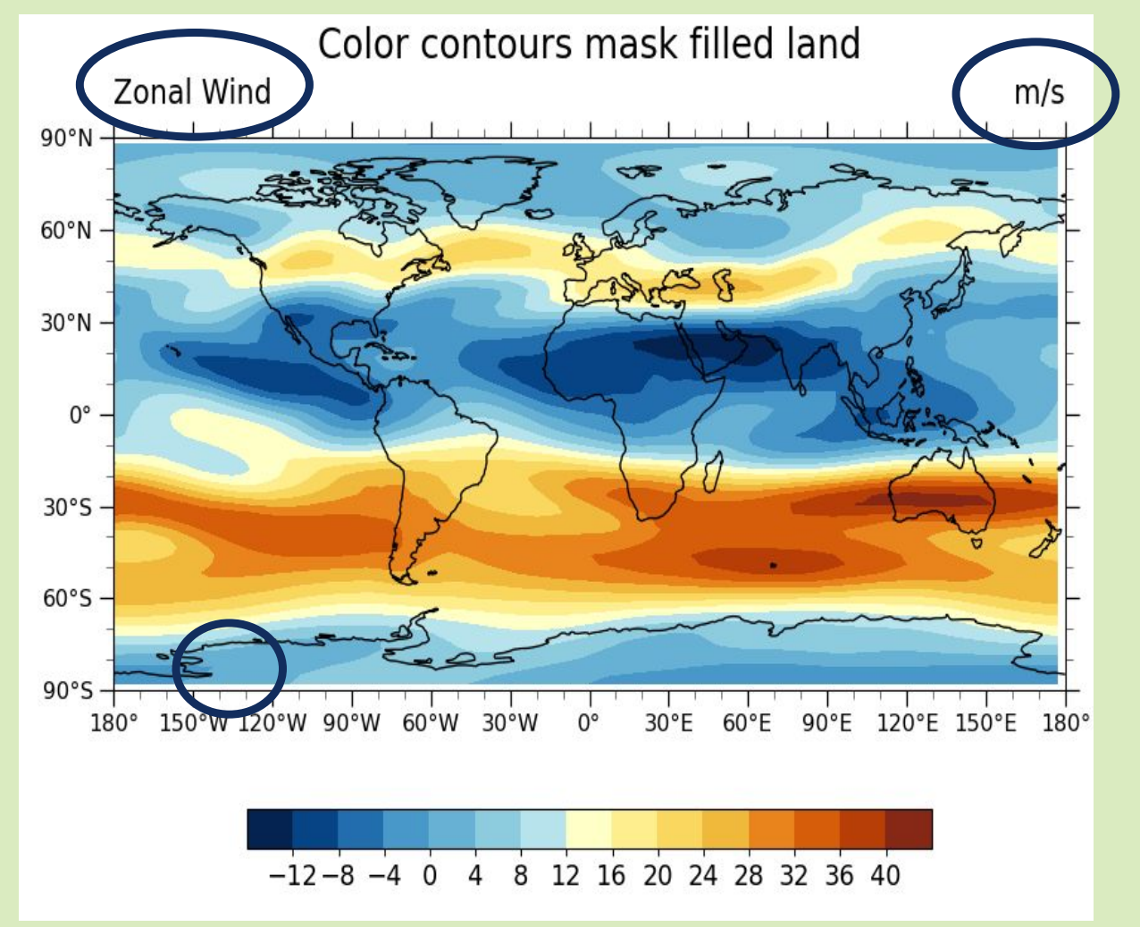
**11+ Utility Functions**

**NCL Plot and Contour Class**

## NCL has stylistic quirks that are not automatically set in Matplotlib

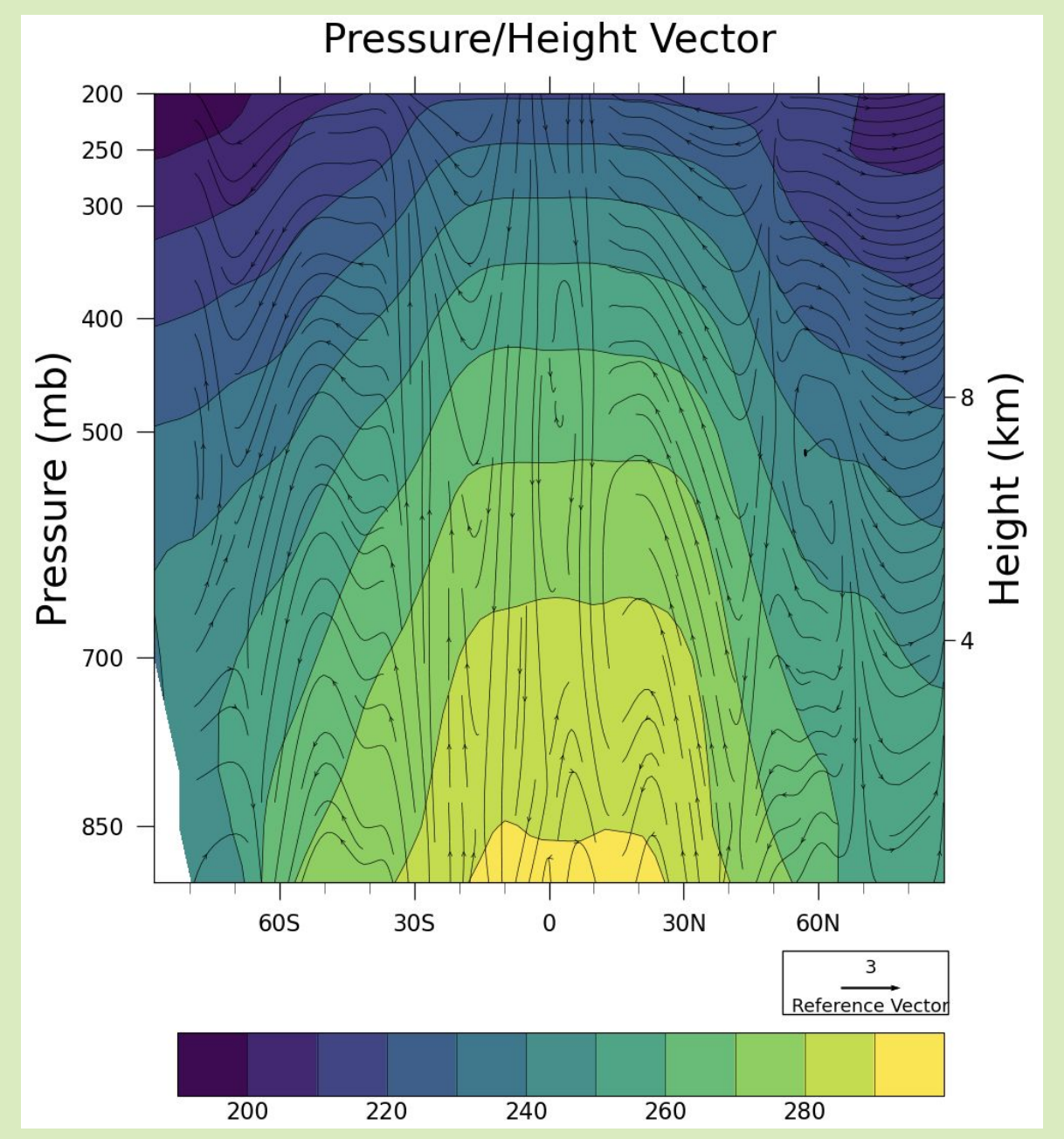
- Left and right subtitles are not built-in features in Matplotlib. Instead, they must be manually placed and sized. Since this is common in many figures, a utility function was included in GeoCAT-Viz to easily add titles.
- Minor ticks, usually about two for every major tick, are a common feature in NCL plots. Again, since this is a common feature in NCL plots that is not embedded in Matplotlib plots, a utility function was made in GeoCAT-Viz.

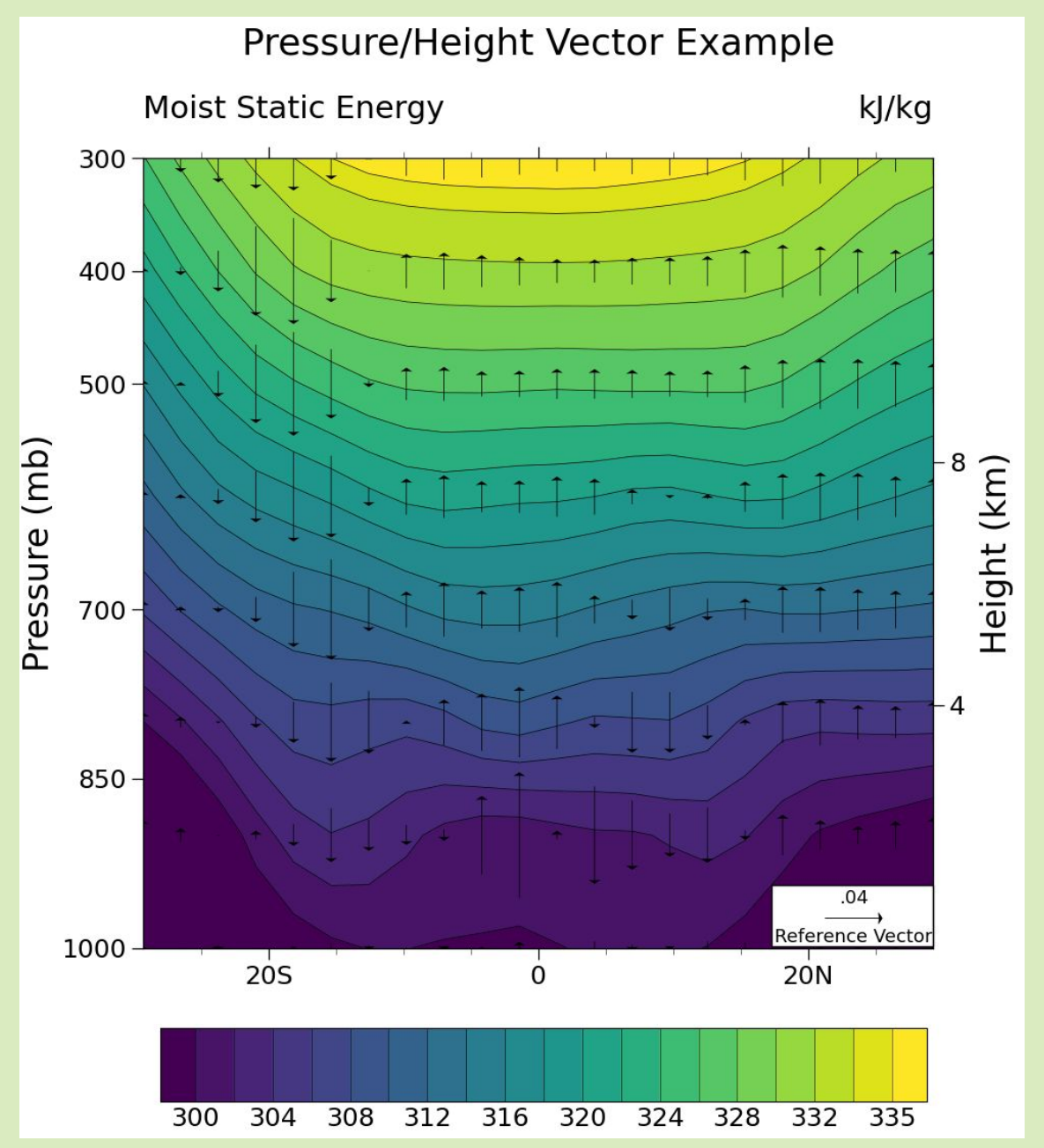
**Repetitive use of utility functions to recreate NCL plots is inconvenient and creates long scripts**



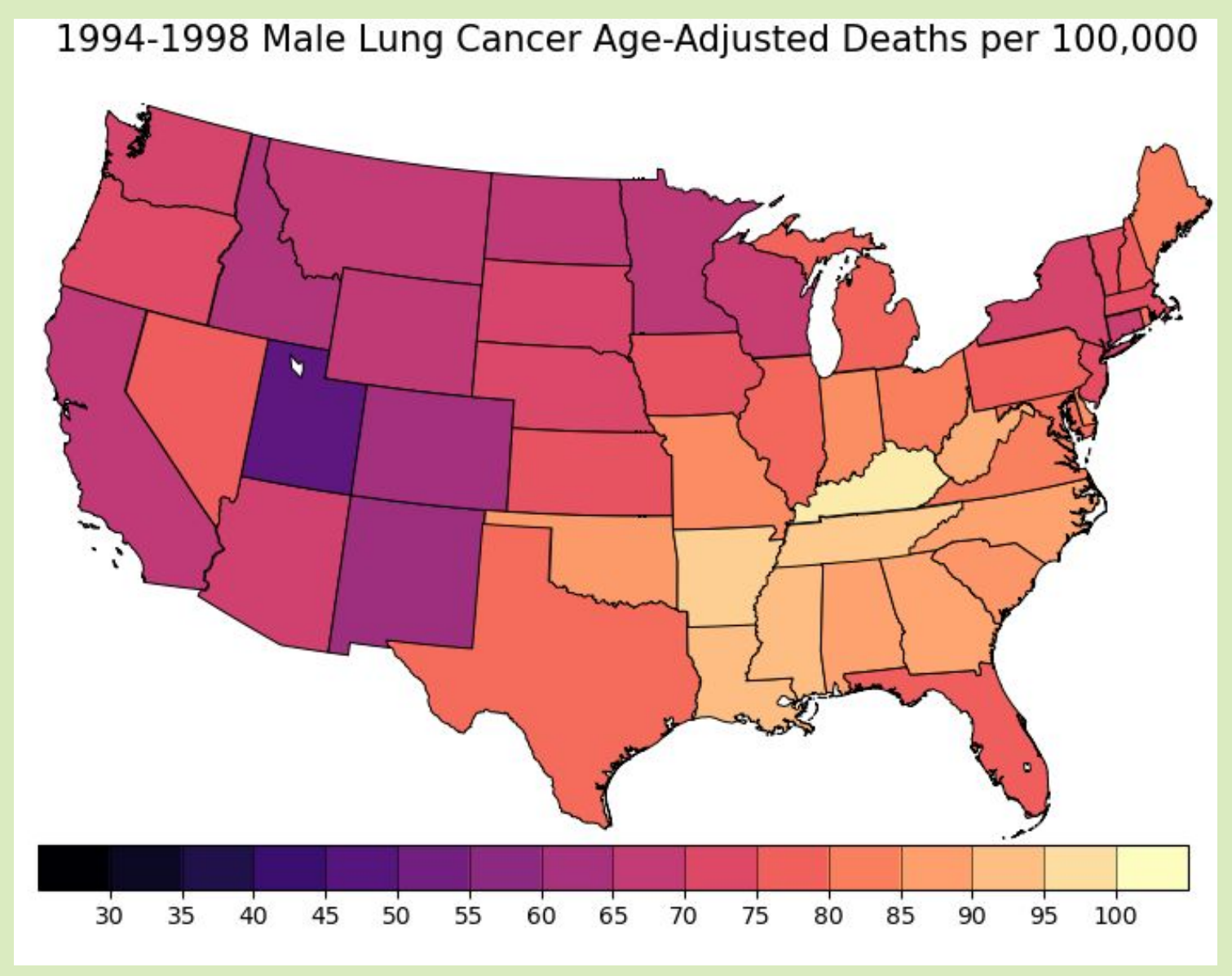
## Examples of Scientific Visualization with Python Ecosystem

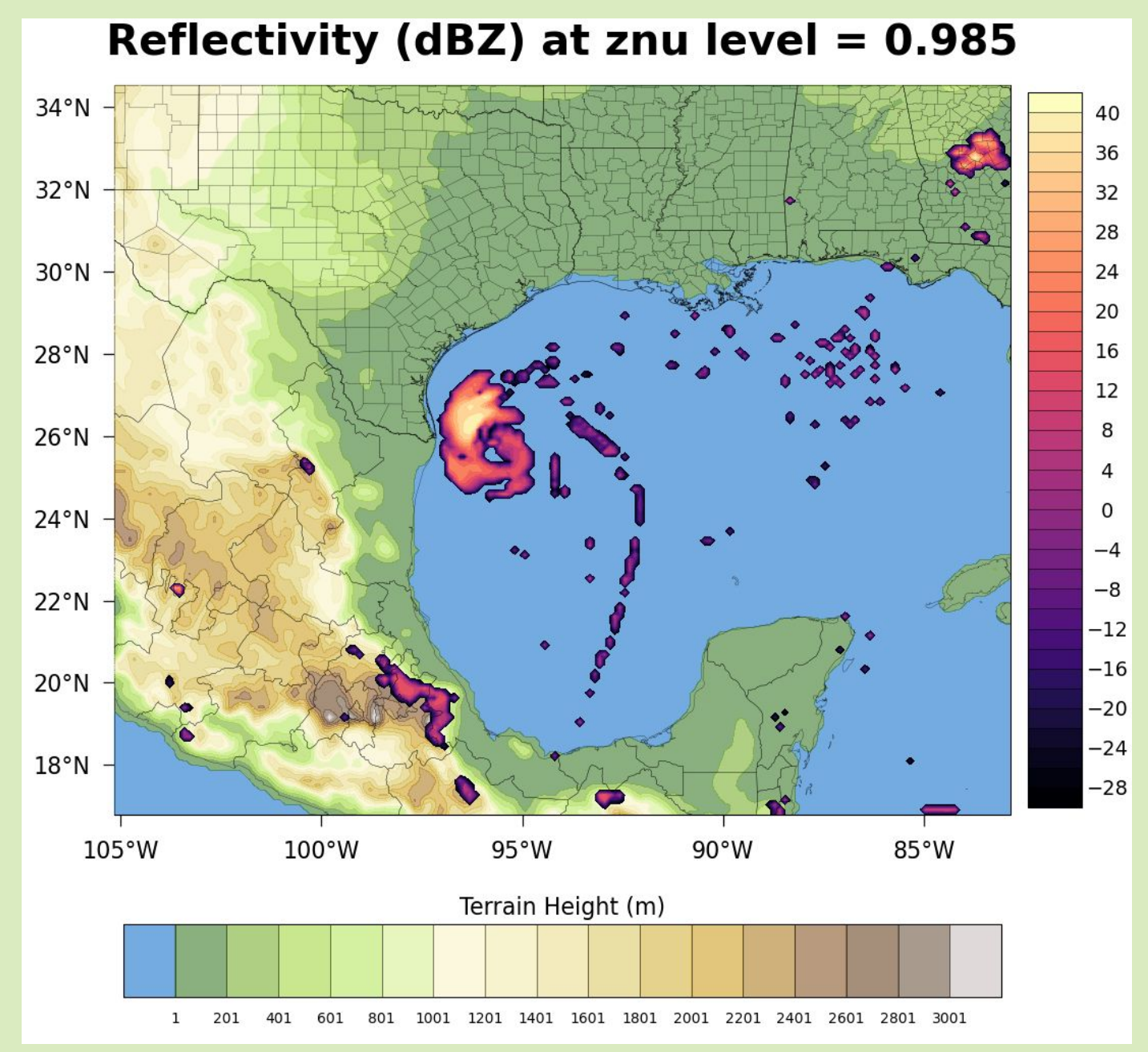
GeoCAT-Comp function interp_hybrid_to_pressure (esccr in NCL) interpolates dataset to user-specified pressure levels



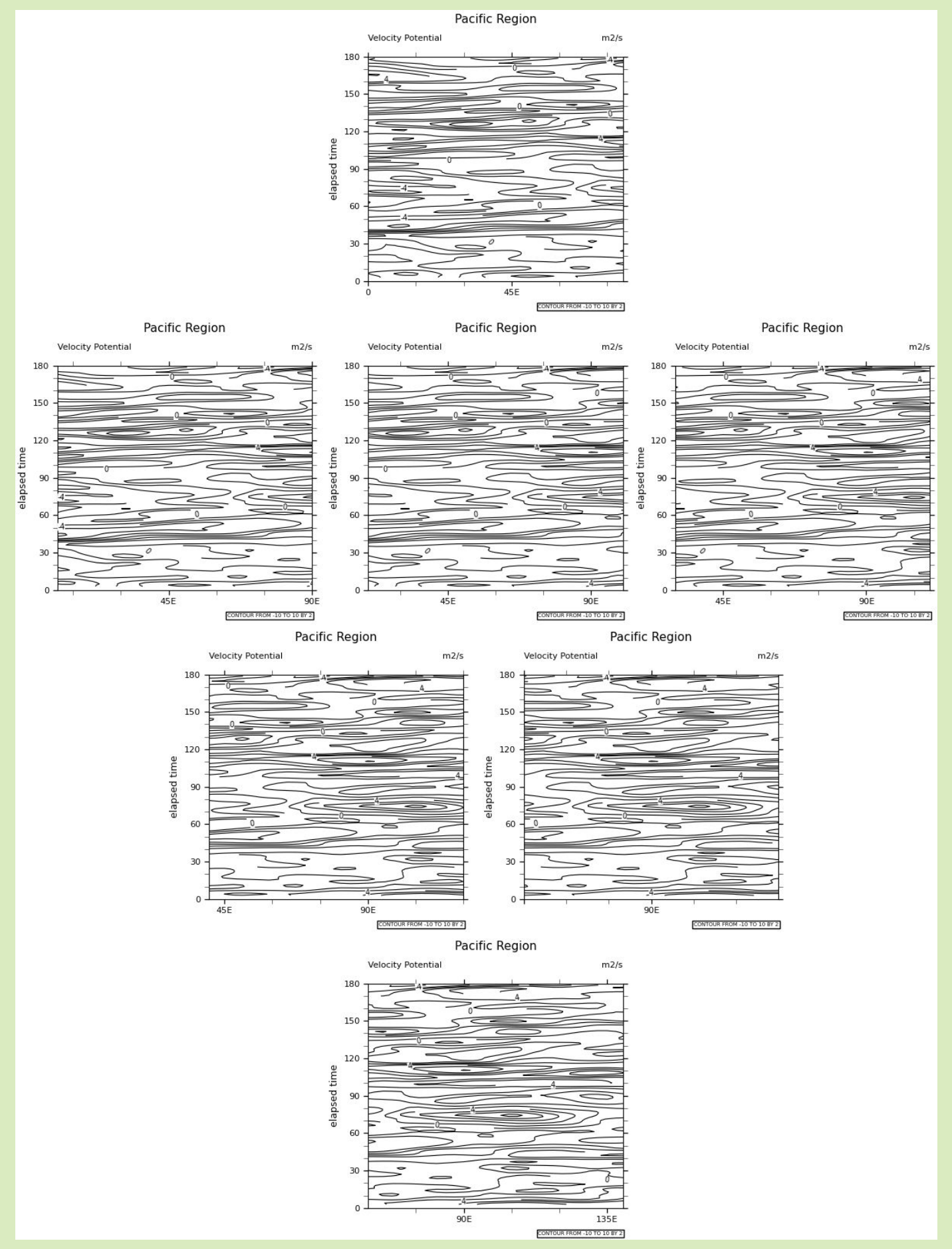


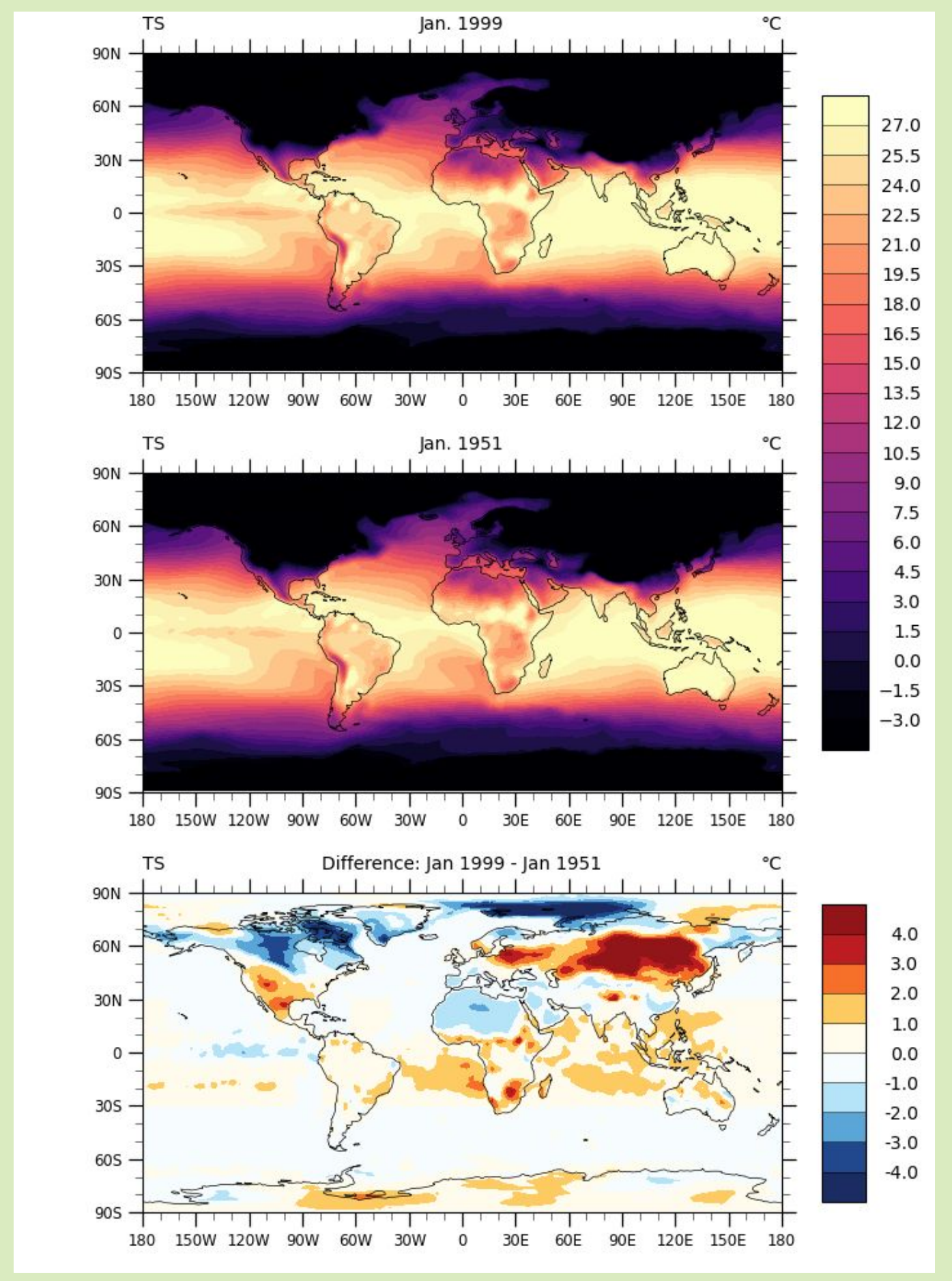
Gradient color map is more accessible than rainbow color map as the brightness difference between levels are visible to people with color blindness and color levels are distinguishable on black and white prints



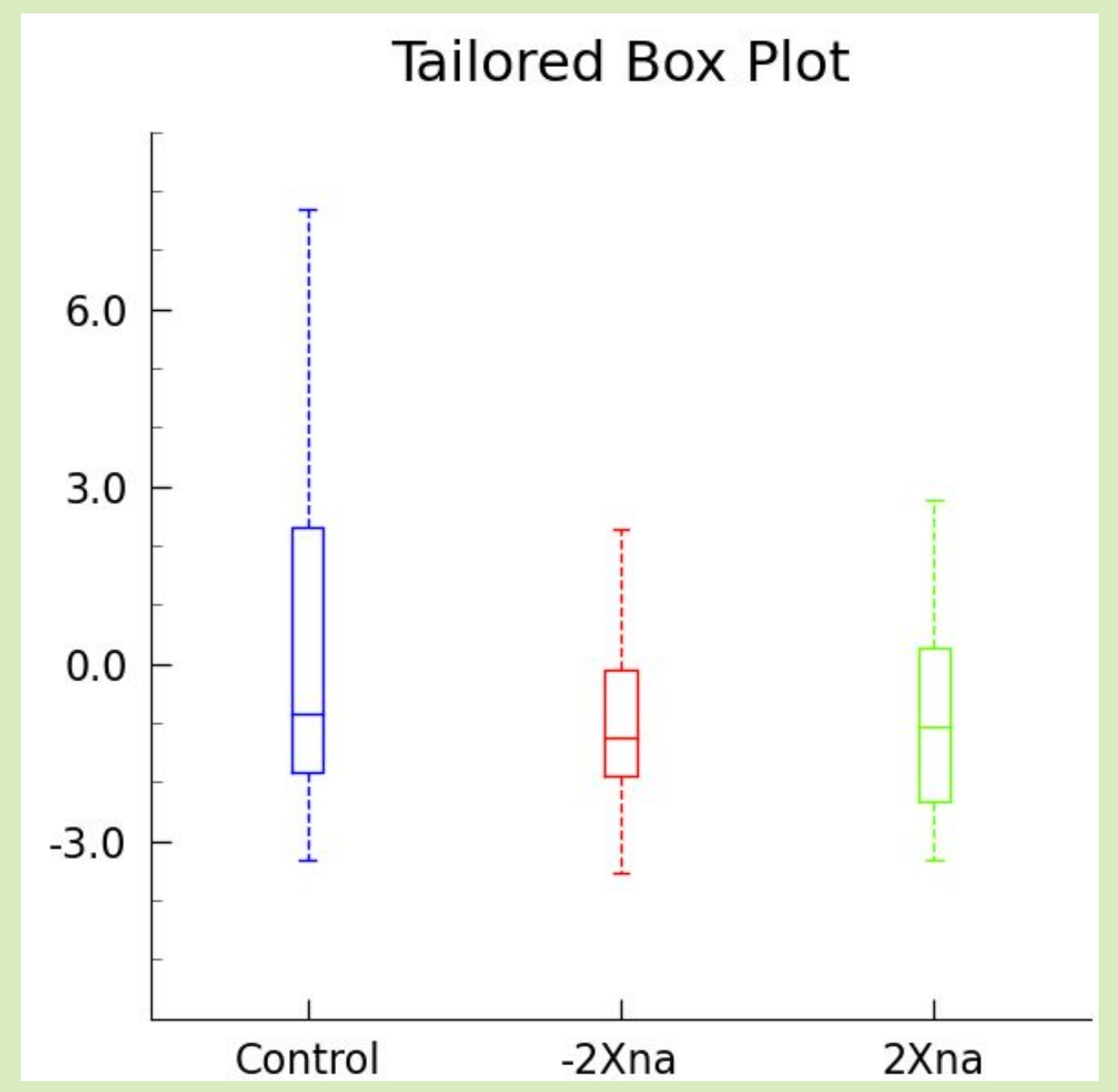


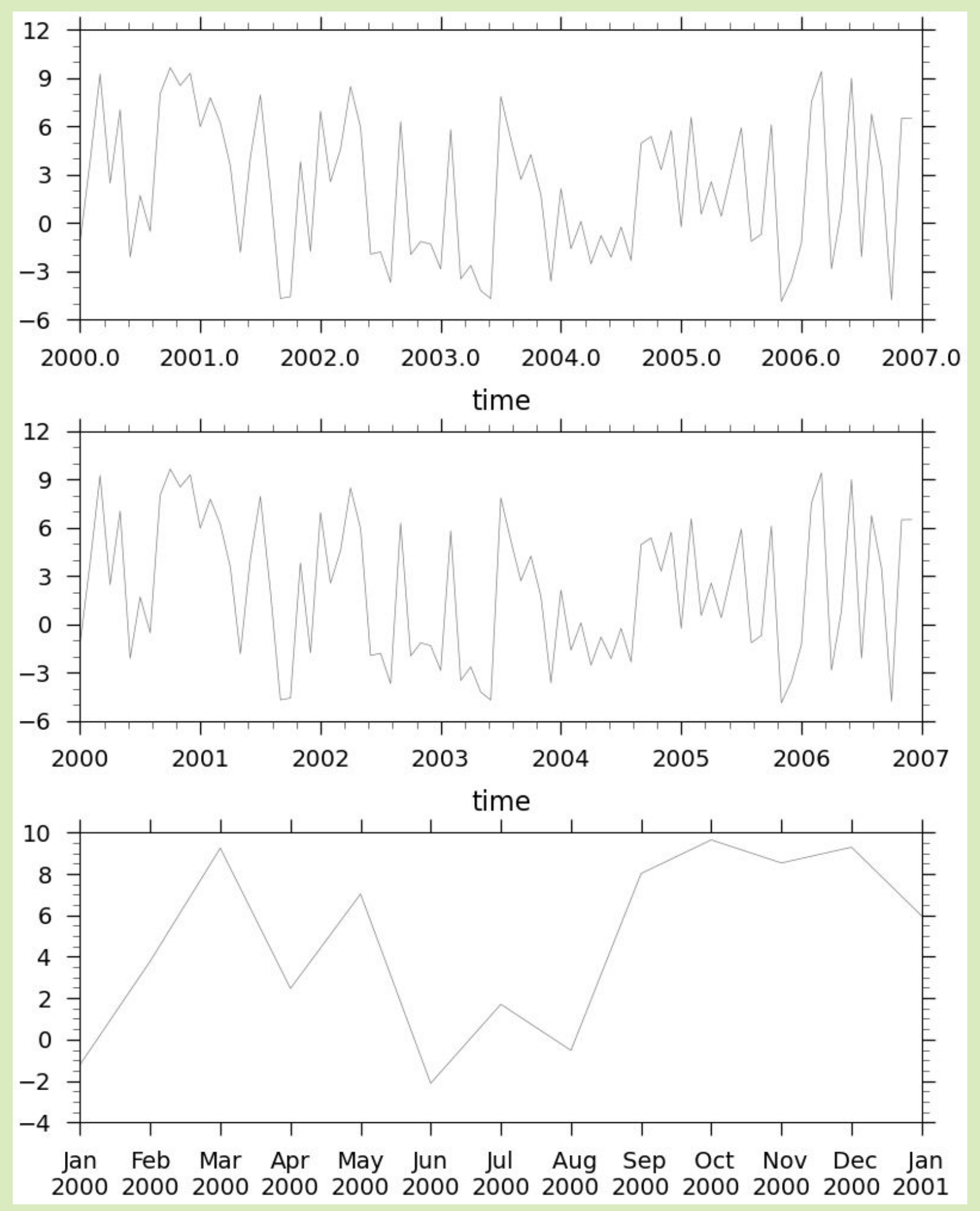
Matplotlib functionalities like GridSpec, Subplot, Inset_axes, axes location are used to configure the positioning of multiple plotting components





Utility functions in GeoCAT-Viz are called to manipulate tick marks formats, font sizes, tick label formats, etc.

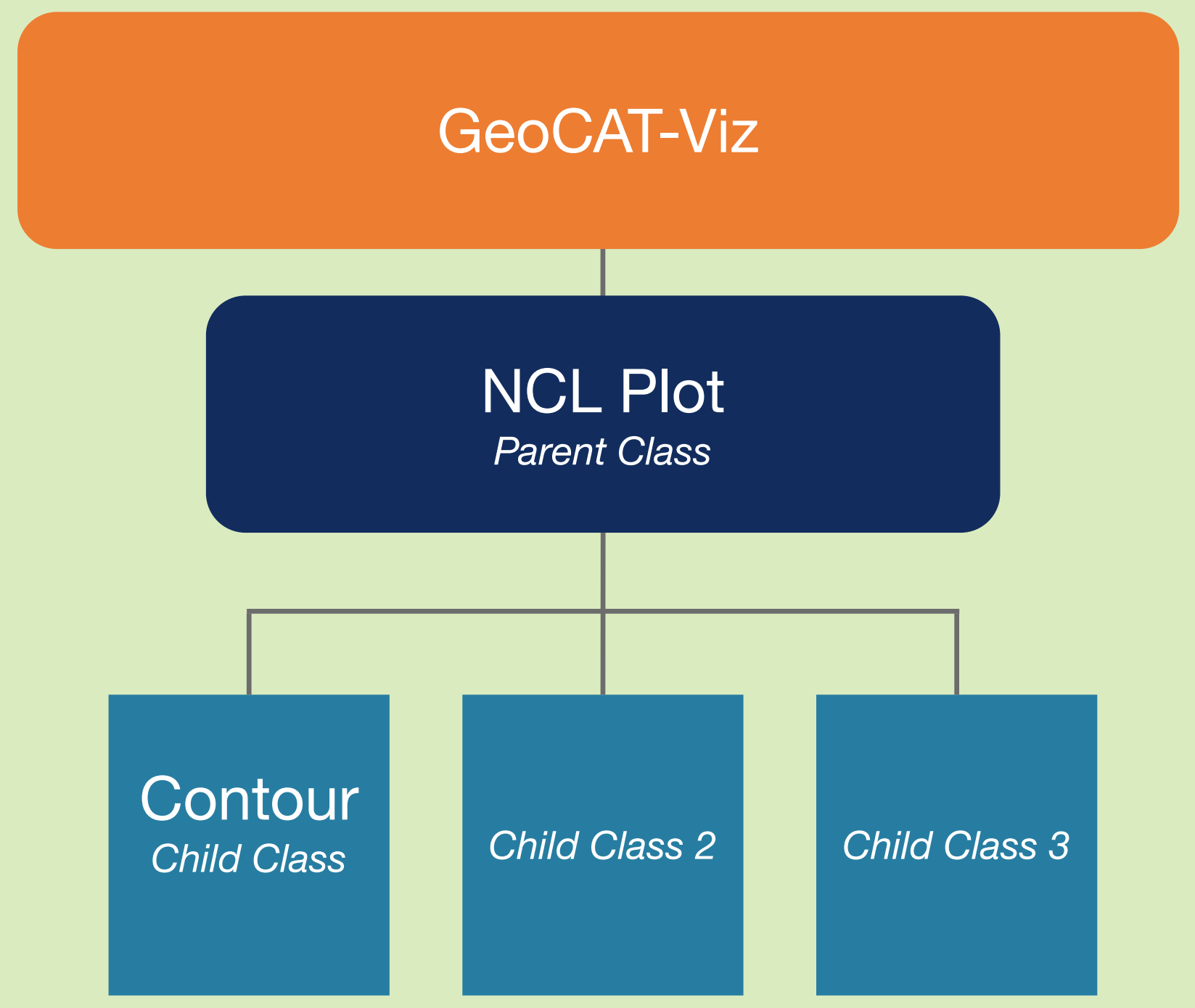




## Transition to NCL Plot and Contour Classes

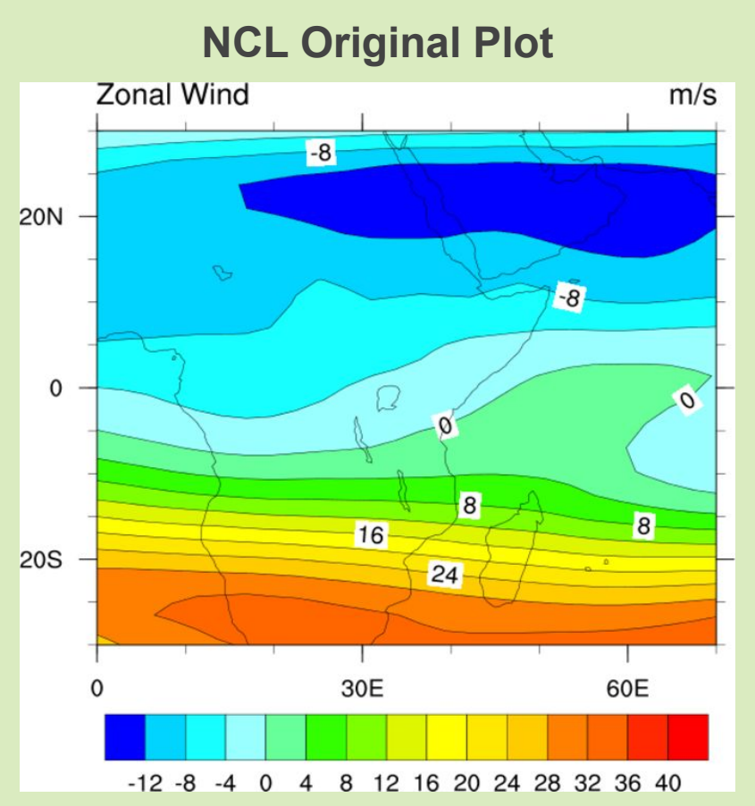
- When replicating plots in Python, it was apparent that basic utility functions were needed to reduce the amount of replicated code between each of the examples.
  - However, as more complex examples were made, multiple utility functions were needed for each example.
  - On average, plots took about 200 lines of code to recreate about 30 lines of NCL.
- This made the transition more difficult as it was hard to convince people to transition to Python when NCL was more efficient for the style they wanted.
- As a result, the NCL Plot and Contour class was created.

**NCL Plot contains basic style and functionalities that are essential across all plots**

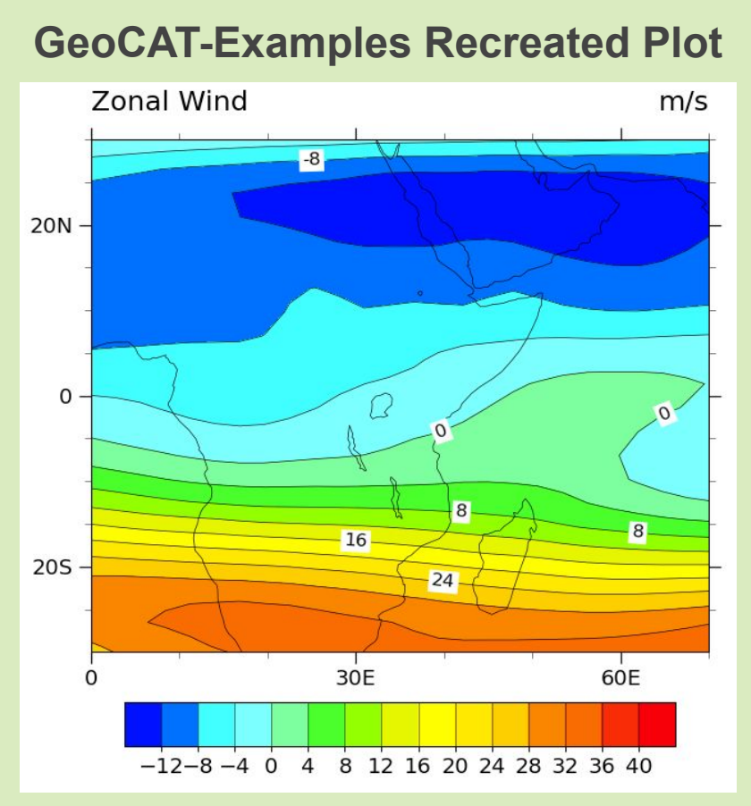


**Contour creates contour plots and adds contour-specific styles, such as contour line labels**

### Figure readability and functionality remains when utilizing the Contour class with less code

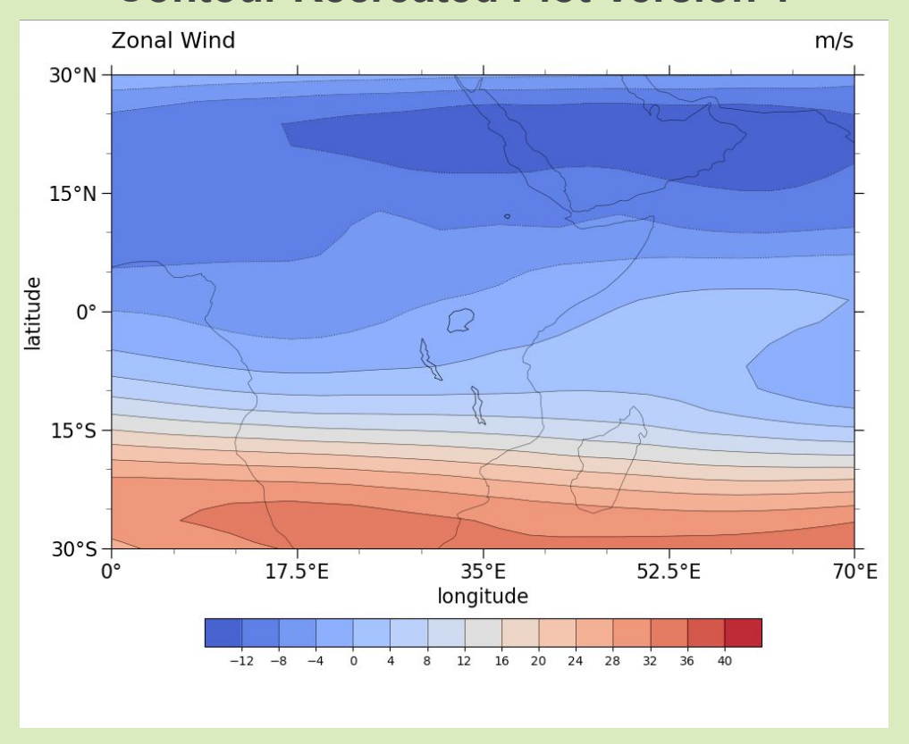


Original NCL Plot: **35 lines**



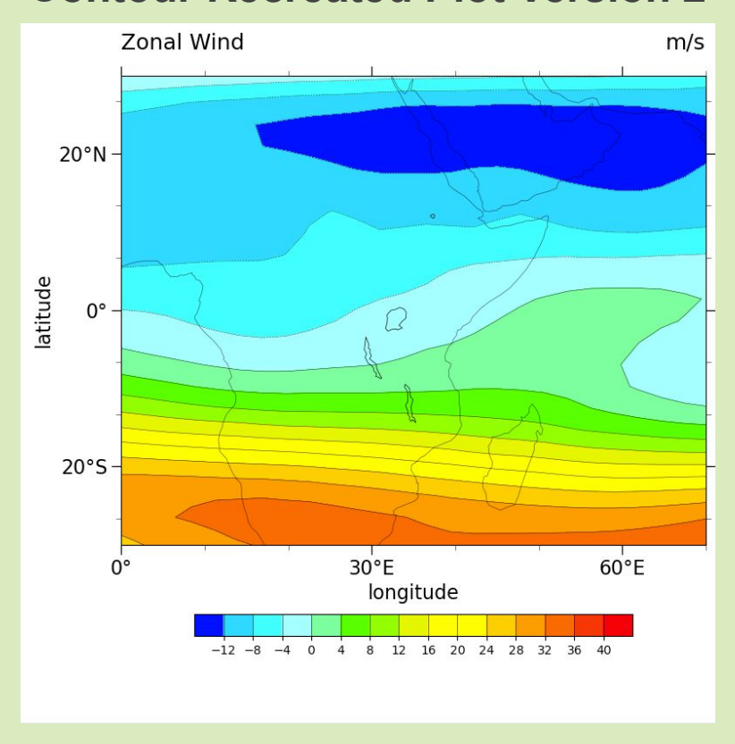
Original GeoCAT-Examples Plot: **108 lines**

Contour Recreated Plot Version 1



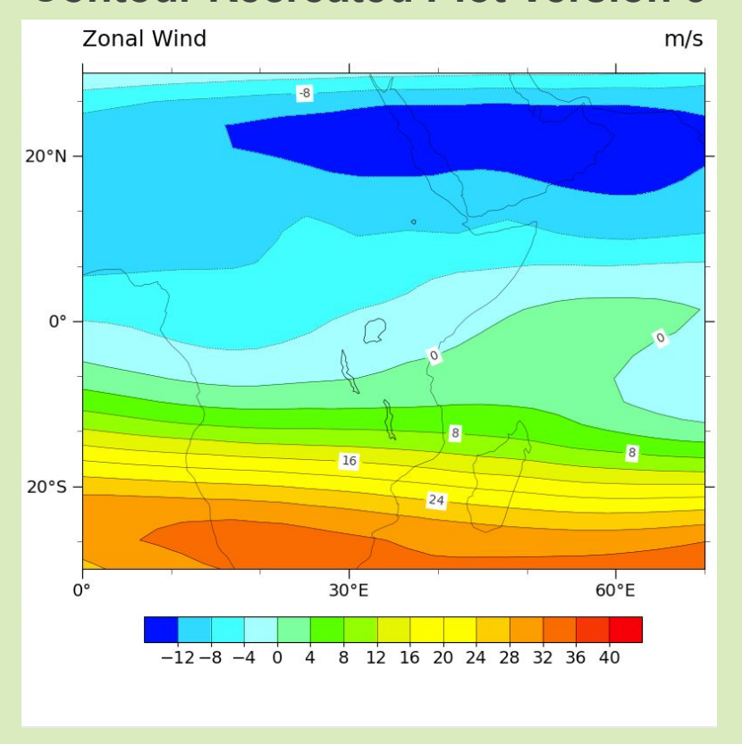
Complete Script: **28 Lines**
Number of Inputs: **7**

Contour Recreated Plot Version 2



Complete Script: **34 Lines**
Number of Inputs: **12**

Contour Recreated Plot Version 3



Complete Script: **47 Lines**
Number of Inputs: **24**

**Adding more inputs dramatically increases the similarity of the plot without significant additions to code length**

Credit to Derek Crowe for providing the poster template