Rebuilding the NCL Visualization Gallery in Python

Claire Fiorino
San Diego State University
SIParCS 2020 Cohort
Facilitate NCAR’s transition from NCAR Command Line Language (NCL) to Python by showing how to emulate NCL features in Python.

- **GeoCAT-Examples** (a gallery of example scripts for different visualizations)
  - Demonstrate all the capabilities of python
  - Document example scripts to make them easy to read

- **GeoCAT-Viz** (a collection of utility functions)
  - Reduce boilerplate code
  - Make visualization scripts easier to write
NCL vs. Python

Rebuilding the NCL Visualizations Gallery in Python

```python
# Change the density with parameter "minDistance"
if minDistance != 0:
    lat_every = 1
    lon_every = 1

# Get distance between points in latitude (y axis)
lat = data["lat"]
latDifference = (float)(lat[1] - lat[0])

# Get distance between points in longitude (x axis)
lon = data["lon"]
lonDifference = (float)(lon[1] - lon[0])

# Get distance between points that are diagonally adjacent
diagDifference = math.sqrt(latDifference**2 + lonDifference**2)

# Initialize ds
ds = data.isel(lat=slice(None, None, None), lon=slice(None, None, None))

if diagDifference >= minDistance and latDifference >= minDistance and lonDifference >= minDistance:
    warnings.warn('Plot spacing is already greater or equal to \'.format(str(minDistance)))

# While diagD
while diagDifference < minDistance or latDifference < minDistance or lonDifference < minDistance:
    # Get distance between points in latitude (y axis)
    lat = data["lat"]
    latDifference = (float)(lat[lat_every] - lat[0])

    # Get distance between points in longitude (x axis)
    lon = data["lon"]
    lonDifference = (float)(lon[lon_every] - lon[0])

    # Get distance between points that are diagonally adjacent
diagDifference = math.sqrt(latDifference**2 + lonDifference**2)

    lat_every += 1
    lon_every += 1

    ds = data.isel(lat=slice(None, None, lat_every), lon=slice(None, None, lon_every))
```

vres@vcMinDistanceF = 0.017
Process of expanding the GeoCAT Gallery

Is there one built-in python function that can achieve the result?

- **NO**
  - Is there a method of doing it with multiple built-in functions?
    - **NO**: Edit module library source code and make an upstream contribution
    - **YES**: Demonstrate the built-in function in an example script

- **YES**: Demonstrate the built-in function in an example script

Will this functionality be used often? Does it require a significant amount of code? Was it hard to figure out?

- **NO**: Demonstrate the use of these built-in functions in an example script
- **YES**: Create a helper function in GeoCAT-viz utility script
• Data Manipulation
  – Methods of altering, or extracting certain numerical features from the data
• Aesthetic
  – Features that do not add any significant meaning to a plot
  – purely for visual appeal
• Plot Manipulation
  – Features that change the way data is visualized on a plot
  – Can affect how data is perceived by viewer, or make a plot easier to understand
Changing, or extracting information from the raw data input of a visualization function

Examples:
- Slicing data
- Smoothing data
- Extracting features:
  - Local extrema
  - Means
  - Maximums/minimums
  - Averages
Data Manipulation: Finding local extrema

- Import data with `xarray`
- Take global gradient of data with `numpy`
- Cluster noisy data with `Scikit-learn` (DBSCAN)
- Find minimum/maximum of each cluster with `numpy`

Geocat-viz.util.py, find_local_extrema()
Basic features (such as changing color, opacity, location items on the plot) are built into these libraries, but others require upstream contributions.

Examples:

- Allowing tick marks on non-rectangular map projections
- Adding multiple arrows to a streamplot graph
Aesthetic Features: Adding Multiple Arrows to Streamlines

One arrow per streamline

Three arrows per streamline

Matplotlib.Streamplot.py
Changes that often require a combination of data manipulation and aesthetic changes.

Examples:

• Changing arrow density in vector plots
• Setting wedge boundaries for non-rectangular map projections
Plot manipulation: Adjusting the spacing on a vector plot

Full vector density

Vector density at 40%

Geocat-viz.util.py, def set_vector_density()
Evolution of the Python Visualization Gallery

Before:

Rebuilding the NCL Visualizations Gallery in Python
Evolution of the Python Visualization Gallery

Rebuilding the NCL Visualizations Gallery in Python
Thank you

Special thanks to mentors John Clyne, Kevin Paul, Julia Kent, Orhan Eroglu, and Michaela Sizemore