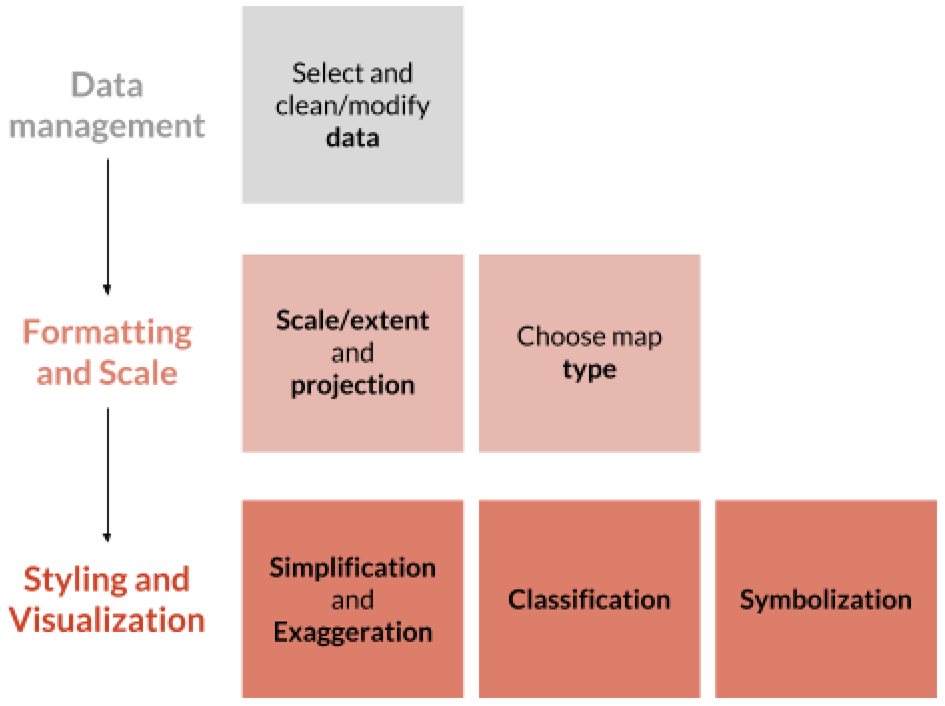
Voting Rights Data Institute

Map Guide

This guide provides an overview of the basic principles of map design and documentation to create maps using the GeoPandas Python library. GeoPandas is an open source Python package that provides the best combination of spatial data analysis and mapping functions within Python. GeoPandas was chosen for this guide due to its accessibility to all Python programmers and simple yet powerful functionality.

Table of Contents

Cartographic Workflow


Data management
Select and clean/modify data
Formatting and Scale
Scale/extent and projection
Choose map type
Styling and Visualization
Simplification and Exaggeration
Classification
Symbolization

1. **Data management** involves choosing the data and attributes that are important to display and conducting any cleaning or enhancement of the data to fit your needs. For our purposes, much of this work will be done in QGIS and *geopandas*. For an introduction to *geopandas* data structures and how to read and write files into *geopandas*, go to the Introduction to GeoPandas section below.

2. **Formatting and Scale** involves defining a map scale (ratio of the map distance to the Earth distance) and map extent (how much of the Earth is shown), and then choosing the right projection based on these choices. Finally, you must decide on a map type that works for your specific data. Information for choosing projection and map type can be found here:
    a. Projections
    b. Map Types

3. **Styling and Visualization** is the final step in the cartographic workflow, but is the most challenging to complete. At its core, cartography is the process of abstracting our complex world by selecting and organizing information that is necessary to communicate a concept to the map reader. Therefore, this final step involves making decisions on how to simplify and exaggerate different aspects of the world to fit your map's purpose. In this guide, information on abstracting is broken down into two sections:
    a. Classification
    b. Style and Visual Hierarchy

Introduction to GeoPandas

GeoPandas is an open source Python package that combines *pandas* data types, *shapely* geometric operations, and several other packages like *matplotlib* for plotting. This guide will provide documentation for cartography with GeoPandas 0.3.0. This section describes the *geopandas* data structures and how to read and write files.

geopandas Data Structures

geopandas implements two data structures, a `GeoSeries` and a `GeoDataFrame`, which are subclasses of pandas `Series` and `DataFrame`.

A `GeoSeries` is essentially a vector where each entry in the vector is a set of shapes corresponding to one observation. Entries in a `GeoSeries` do not need to be all the same geometry type, yet some operations will fail if not. *geopandas* has three basic classes of geometric objects (which are actually *shapely* objects): Points / Multi-Points, Lines / Multi-Lines, Polygons / Multi-Polygons.

Click on the corresponding link for more information on `GeoSeries`:

- Attributes
- Basic methods
- Relationship tests

A `GeoDataFrame` is a tabular data structure that contains a `GeoSeries`. Any of the attributes, calls or methods described for a `GeoSeries` will work on a `GeoDataFrame` – effectively, they are just applied to the "geometry" `GeoSeries`.

The most important property of a `GeoDataFrame` is that it always has one `GeoSeries` column that holds a special status. This `GeoSeries` is referred to as the `GeoDataFrame`'s "geometry". When a spatial method is applied to a `GeoDataFrame` (or a spatial attribute like `area` is called), this commands will always act on the "geometry" column. The "geometry" column – no matter its name – can be accessed through the `geometry` attribute (`gdf.geometry`), and the name of the `geometry` column can be found by typing `gdf.geometry.name`.

Reading and Writing Files
geopandas can read almost any vector-based spatial data format including ESRI shapefile, GeoJSON files and more using the command `gpd.read_file()`, which returns a GeoDataFrame object.

GeoDataFrames can be exported to many different standard formats using the `GeoDataFrame.to_file()` method. For a full list of supported formats, type `import fiona; fiona.supported_drivers`.

Projections

Every map is wrong! When trying to plot a three-dimensional geoid, such as the Earth, on a two-dimensional surface, there is always distortion. However, choosing an appropriate coordinate system/projection allows you to preserve the geographic elements that are important to your specific map. A coordinate system is a reference system used to represent the locations of geographic features and there are two primary kinds:

Geographic and Projected Coordinate Systems

- **Geographic (GCS)**: method for describing the position of a geographic location on the earth's surface using spherical measures of latitude and longitude. A point is referenced by its longitude and latitude values, yet length or area calculations using these values are meaningless.

- **Projected (PCS):** defined on a flat, two-dimensional surface. Unlike a GCS, a PCS has constant lengths, angles, and areas across the two dimensions. You'll generally want a PCS for thematic maps.

To choose a projection for a specific project, there are two things to keep in mind:

1. *Does this map require a certain spatial property to be preserved?*

   Certain projection types preserve specific spatial properties. For example, conical projections preserve angles, planar projections preserve distances, and equal-area projections, unsurprisingly, preserve areas. Here is a great resource for choosing a projection from Hunter College at CUNY. Most maps for redistricting will want to preserve area so equal-area projections are usually going to be the best choice.

2. *What is the scale of your area of interest?*

   Every projection is designed to minimize distortion at a certain range of scales. For our purposes, most maps will either be of the entire United States or of a specific region, state, or district. At the national scale, the best and most-commonly used equal-area projection is the Albers Equal Area Conic. For a larger scale map of a specific state or district, the best choices are Universal Transverse Mercator (UTM) Zones, which define zones along specific longitude ranges, or the State Plane system, which has pre-defined projections for states and portions of states.

More information: What are map projections? - in-depth overview from Esri

Projections in GeoPandas

GeoPandas uses Coordinate Reference Systems to tell Python how the coordinates associated with GeoSeries and GeoDataFrames are associated with the Earth. CRS are like PCS and are referred to using codes called proj4 strings. You can find the codes for most commonly used projections from here and clicking on Proj4.

GeoPandas accepts representations of CRS in many different forms. For the common WGS84 GCS projection, the proj4 representation of this projection can be: `"+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs"`, yet common projections can also be referred to by EPSG codes, so this same projection can also called using the proj4 string `"+init=epsg:4326"`.

For reference, a few very common projections and their proj4 strings:

WGS84

- `"+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs"` or `"+init=epsg:4326"`

UTM Zone 33 (North)

- `"+proj=utm +zone=33 +ellps=WGS84 +datum=WGS84 +units=m +no_defs"`

UTM Zone 33 (South)

- `"+proj=utm +zone=33 +ellps=WGS84 +datum=WGS84 +units=m +no_defs +south"`

To set a projection: `my_geoseries.crs = {'init' :'epsg:4326'}`

Setting a projection is only necessary when your data has x-y data, but no associated coordinate system. Data that is loaded from a reputable source (e.g. using the `from_file` *command) should always include projection information.*

To re-project: `my_geoseries = my_geoseries.to_crs({'init': 'epsg:3395'})`

Map Types

Maps are generally broken down into two main groups: **Reference** and **Thematic** maps.

Reference maps emphasize the location of spatial phenomena, such as countries, cities, rivers, etc. The best example of reference maps are products like Google Maps. Reference maps are useful for showing where things are, but not the relationship between phenomenon.

By contrast, **thematic** maps emphasize the spatial pattern of geographic attributes or statistics about places and relationships between places. Thematic maps will be the most useful for the purposes of VRDI and MGGG and can be further broken down into several categories:

- Areal features:
    - Choropleth - areas are shaded in proportion to the measurement of the statistical variable being displayed
- Point features:
    - Dot density - shows geographic patterns by using a dot symbol to show the presence of a feature or a phenomenon
    - Proportional and graduated symbols - uses size to represent differences in the magnitude of a discrete, abruptly changing phenomenon
    - Heat maps - method of showing the geographic clustering of a phenomenon

Creating Maps in GeoPandas/Python

geopandas provides a high-level interface to the `matplotlib` library for making maps. Mapping shapes is as easy as using the `plot()` method on a `GeoSeries` or `GeoDataFrame`. In general, any options that work with pyplot in matplotlib work with `plot()`.

Choropleth

- Use the `plot()` command with the `column` argument set to the column whose values you want used to assign colors.
- Modify the colors used by plot with the `cmap` option (full list of colormaps).
- Change the scale with the `scheme` option (if you have `pysal` installed). The `scheme` option can be set to 'equal_interval', 'quantiles' or 'percentiles', and more. See the Classification Schemes section for further details.

  Example call: `world.plot(column='gdp_per_cap', cmap='OrRd', scheme='quantiles')`

Dot Density

- Use the `plot()` command with `GeoDataFrame` point data

Proportional and graduated symbols

- This guide from SUNY Buffalo has a great, detailed workflow on how to create graduated symbol maps using *geopandas*

Heat maps

- This Jupyter Notebook guide has good information on how to generate heat maps from `GeoDataFrame` point data using *matplotlib* and *numpy*
- This program, heatmap.py, from Seth Golub, is a tool that creates a heat map from point data such that data density is shown using brightness. This program does not use *geopandas*, but simply requires lat-lon data and can be run from the command line.
- This article from EatSleepData describes how to generate a geographical heat map using *gmplot*, which plots a heat map over Google Maps.

Style and Visual Hierarchy

The most important aspect of map design to keep in mind is **visual hierarchy**, which is the organization of design such that some things seem more prominent and important and others less so. The visual hierarchy of a map influences how people read it and perceive the relative significance of its elements.

These elements include the data or information associated with the map, any labeling or explaining text, geographic markers, and other map features like a title or legend. It can be helpful to actually write out an ordered list of your map elements by where it should be in the visual hierarchy. Below is a list of tips and helpful resources for designing different elements of a map:

Color: *hue, value, saturation*

- Choose colors for individuals with visual impairments such as color-blindness (i.e. don't use red-green color schemes)
- Make sure the logic of your colors relates to the logic of your map elements (i.e. use blue for water, green for open spaces, red/blue for party affiliations)
- Related to the above, be aware of possible associations with your colors (i.e. do not use red/blue for political maps if you do not mean Republican/Democrat)
- For nominal or unorderable data:
    - different hues, constant saturation and value
- For orderable categories or numerical data:
    - Sequential color scheme: single or multi-hue, but they are dominated and ordered by differences in lightness/saturation
    - Diverging schemes: should only be used when your data has a natural mid-point such as a zero (e.g., positive and negative change/growth) or if you want to compare places to something like the national average
- Other (very) helpful **color** resources
    - Axis Maps Color Guide - great overview on using color on maps
    - ColorBrewer - popular web tool for guidance in choosing choropleth map color schemes, based on the research of Dr. Cynthia Brewer
    - Adobe Color - create color schemes with the color wheel or browse thousands of color combinations
    - Paletton - color scheme designer with options for complementary, monochromatic, adjacent, triad, and more.

Text: *font and font style, size, positioning*

- Choose fonts that have a variety of styles, such as **bold, extra bold,** *italic*, ***bold italic*** light, etc to differentiate between different kinds of features on your map
- Fonts that have both a serif and sans serif version are good for creating a unified look on your layout. People often use one version for the map body and one for the explainer text outside the map. Here are examples of serif and sans serif fonts:
    - **Serif** - font type with a small line attached to the end of strokes
        - PT Serif (Google Fonts)
        - Merriweather (Google Fonts)
        - Aver (Free)

        - Lora (Google Fonts)
    - **Sans serif** - font type without serifs
        - Open Sans and Open Sans Condensed (Google Fonts)
        - PT Sans and PT Sans Narrow (Google Fonts)
        - Ubuntu and Ubuntu Condensed (Google Fonts)
        - Oswald (Google Fonts)
        - Merriweather Sans (Google Fonts)
- Positioning
    - Prioritize the position of point feature labels: 1) above and to the right, then 2) below and to the right, then 3) above and to the left, then 4) below and to the left. Positioning directly above, below, or to the sides is not preferred.
    - Visually center and increase the letter-spacing of labels within area features to reinforce their size/shape.
- Size
    - Distinguish ranked categories by at least two points when label sizes are small.
    - Labels should not be smaller than around 6-7pts for print maps / 9-10pts for maps displayed on screen.
- Style
    - Use uppercase to label area features
    - Letter spacing with uppercase can be helpful within area features to fill the space
    - Categorize cultural and physical features using sans serif and serif fonts
- Other helpful resources
    - Axis Maps Labeling and Text Hierarchy Guide - great overview for labeling and text hierarchy in cartography
    - Google Fonts - listing of available web fonts, can help with pairing fonts
    - Type in Maps - helpful guide from Penn State on characteristics of labeling

Other map elements: *title/subtitle, legend, credits, explanatory text, inset & locator maps, white space*

- The list of elements above are some of the map features you should think about including on any map layout. The most important thing to remember is that the map is **only one part of your layout**; other elements like legends, titles, explanatory text must fit within your visual hierarchy and overall design.
- **Titles/subtitles** are essential elements of any map to give map readers a quick understanding of what your map shows. Make sure the styling of your title does not make it stand out too much.

- **Legends** are necessary on most maps to give readers a guide to your classification scheme and an understanding of what different features refer to.
- All map layouts need some **white space** to create balance.
- Add enough **explanatory text** that is necessary to explain what the map shows, but too much text will be overwhelming for a reader.
- **Inset or locator maps** can be useful to give readers geographic context for a zoomed-in map, such as for a congressional district you could show where it lies within the state.
- Always add **credits** to your map layout, preferably on the bottom in one of the corners. The credits should be the lowest element on your visual hierarchy.

Classification Schemes

Choropleth and graduated symbol maps require some method of **data classification** whereby the observations in a dataset are grouped into data ranges or classes. It is much easier for map readers to interpret data if there are just a few well-defined classes. The recommended number of classes is generally between three and seven.

There are many different classification schemes to choose from, and there are advantages and drawbacks to each of them. Different choices of classification scheme can drastically affect how people interpret maps, so it is important to select one that clearly communicates the objective of the map.

Equal Interval: *divide the classes into equal groups*

- Advantages
  - Highlights distribution of observations throughout the entire data range
  - Highlights outliers in data
- Disadvantages
  - Can sometimes break up similar observations into different classes, meaning that it can sometimes mask clusters in the data
  - The number of observations in each category can vary greatly, even including some potentially empty classes

Quantile (Equal Count): *arrange classes so that they each have the same number of observations (quintile = five classes, quartile = four, etc.)*

- Advantages
  - Allows you to easily visualize the top and bottom 20% of the data
- Disadvantages

    - Can mask outliers and break up clusters
    - The ranges of classes can greatly vary

Natural Breaks (Jenks): *minimize variation in each group by placing class breaks between large gaps in observations*

- Advantages
    - Allows for highlighting outliers and keeping clusters together
    - Minimizes variation within groups
- Disadvantages
    - Can be subjective

Standard Deviation: *Each standard deviation becomes a class*

- Advantages
    - Shows the statistical origin of the data
- Disadvantages
    - The raw classes need clarification for audience (i.e. What is the mean?)

More information: The Basics of Data Classification - great overview of data classification techniques from Axis Maps

Classification Schemes in GeoPandas

The way color maps are scaled can be manipulated with the `scheme` option (if you have `pysal` installed, which can be accomplished via `conda install pysal`). The `scheme` option can be set to 'equal_interval', 'quantiles' or 'percentiles', and many more. See the PySAL documentation for further details about these map classification schemes. Here is a Jupyter Notebook with much more information about using PySAL classification with *geopandas*.

Example call: `world.plot(column='gdp_per_cap', cmap='OrRd', scheme='quantiles');`

Interactive Maps

Interactive maps are high risk, high reward. Highly interactive maps can be too difficult for new users, yet they can also allow users to learn and explore at their own pace. It is always important to test designs on multiple people with varying degrees of technical expertise to see how they interact differently with the layout. Interactive map design should reflect what the cartographer is trying to convey (beauty vs. function) and never sacrifice visual hierarchy or great styling for interactivity.

If you are wondering whether you should make your map interactive, here is a great starting guide from Axis Maps: should a map be interactive?

Here are some general tips for designing effective interactive maps:

Simplify the interface

- Panels should be the minimum possible size so they don't clutter the space
- Don't innovate to the point where users don't know what to do
- Use panel opacity and margins to help them blend with your basemap and look cohesive
- Leave space for the maps to be used on mobile devices
    - This means avoiding permanent panels

Pop-ups and markers

- Good marker examples:
    - Flaticon
    - The Noun Project
    - Font Awesome
- Keep pop-ups concise
- Only one pop-up on the screen at a time (generally)
- Cluster markers for better readability
- Use a legend when you have multiple types of markers
- Remove unnecessary padding and margins from popups
- Use a slight shadow to "lift" pop-ups off the map

Other resources

Principles of interactive map design by Victor Gerard Temprano
The guide to map design by Mapbox

Great examples of interactive maps

A Tale of Two Cities by Ilia Blinderman and Amber Thomas
The Atlas of Redistricting by FiveThirtyEight
Mapping America's rental housing crisis by Urban Institute
Conflict Urbanism: Colombia by Center for Spatial Research

Interactive Maps in GeoPandas/Python

Geopandas does not have interactive functionality built-in to the package, but there is a package called *Bokeh* that generates interactive visualizations and makes use of *geopandas.* Here is good documentation of *Bokeh* and examples of how to use it for interactive mapping with *geopandas.*

Compiled by Zach Levitt, Ethan Ackerman, Ruth Buck, Katie Jolly, and Katya Kelly
Voting Rights Data Institute, July 2018