ExtendedComponent

We propose that glue add a new ExtendedComponent(Component) that describes a set of shapes corresponding to single observation. ExtendedComponents describe an extant on one or more other Components in the dataset (parent_components— which may be CoordinateComponents or regular Components), and should contain references to those parent_components. Data loaders would be responsible for parsing input data files into the appropriate components and individual Viewers would be updated to support plotting ExtendedComponents. ExtendedComponents should perhaps be hidden from the user and only used by Viewers when plotting the parent_components and transforming ROIs to SubsetStates.

Motivating Use Cases

- Geographic regions (e.g. https://github.com/jfoster17/glue-map)
- Regions-of-interest in astronomy images (e.g. https://github.com/glue-viz/glue-regions)
- Genomic regions-of-interest (e.g. genes, which have a finite extent on the genome)
- Cell boundaries in microscopy images

Dimensionality

The dimensionality of an ExtendedComponent may be different from the dimensionality of the other components in a Data object. For instance, in a table of information about the U.S. States, the dimensionality of the Data object is 1D, although the ExtendedComponent describes 2D objects (or a 1D list of 2D objects in the case of Hawaii, which is multiple polygons). ExtendedComponents should return a shape and ndim that is consistent with the dataset they belong to, rather than their actual shape and ndim.

Selections/Subsets

ROI Translation: contains center, intersects, or fully contains?

Using ExtendedComponents requires some re-thinking of how to translate an ROI to a SubsetState. The basic question is how to handle a ROI that partially intersects an ExtendedComponent. First, if we require that every ExtendedComponent is associated with point-like parent_component(s), then we could simply apply the ROI to those components. in glue-map we use Shapely's representative_point() as the point-like components associated with an extended region. We could also include a region in an ROI if the region (a) intersects with the ROI or (b) is fully contained within the ROI. We could expose these options to the user as different selection tools or as different modes (modes are hard, and we already ask the user to remember which type of boolean mode they are in for subset creation).

In glue-map we have two different tools to define ROIs. The user can draw a rectangle, which defines a rectangular ROI in lat/long or the user can click on individual regions, which defines an ROI which is the combined shape of all regions selected (and thus automatically contains all points within those regions).

Transformation from regions to subsets

The other common operation we need to support is transformation from a selection of regions (as rows in a table) to a SubsetState that covers their extent (i.e. the union of their shapes). For example, if the user has point-like data on cities and region-like data on states, then there needs to be a way for a subset of states (whether from a table viewer or a scatterplot of state properties) to be translated into a SubsetState representing that geographic region so that the susbset can be applied to cities. This could be a manual process (either using a common toolbar item or a menubar/contextual menu selection) or it could be automatically handled through a new kind of link — perhaps an EncompassesLink that tells glue that subsets of these regions should propagate subsets to these points.

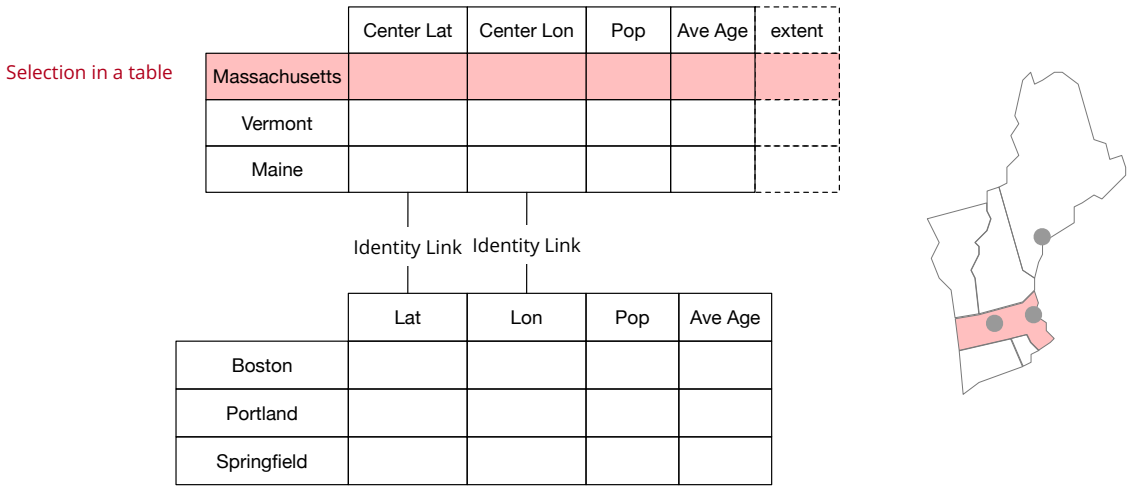
EncompassesLink

An EncompassesLink potentially gets complicated; consider the following cases where we have states (regions) and cities (points).

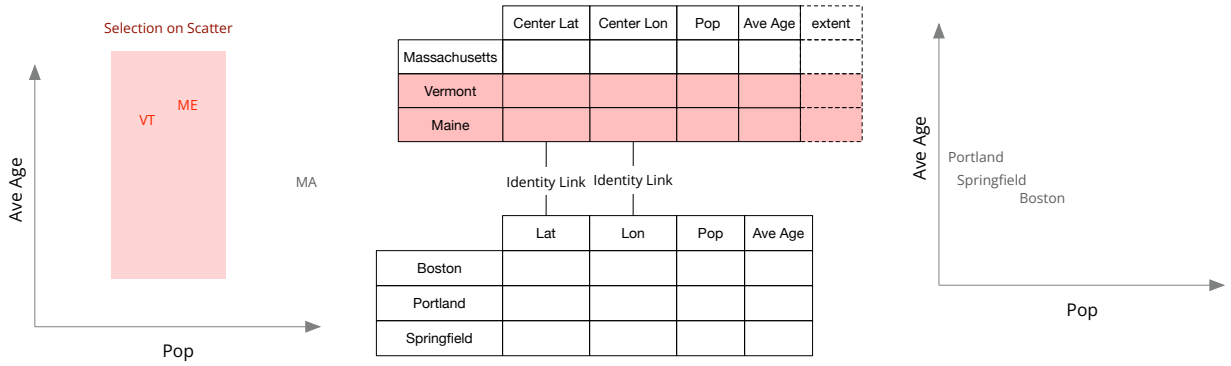
Note: in the following examples, Springfield is Springfield, MA, not Springfield VT. I should have chosen a different city, but I'm not remaking all the figures :)

Case A:

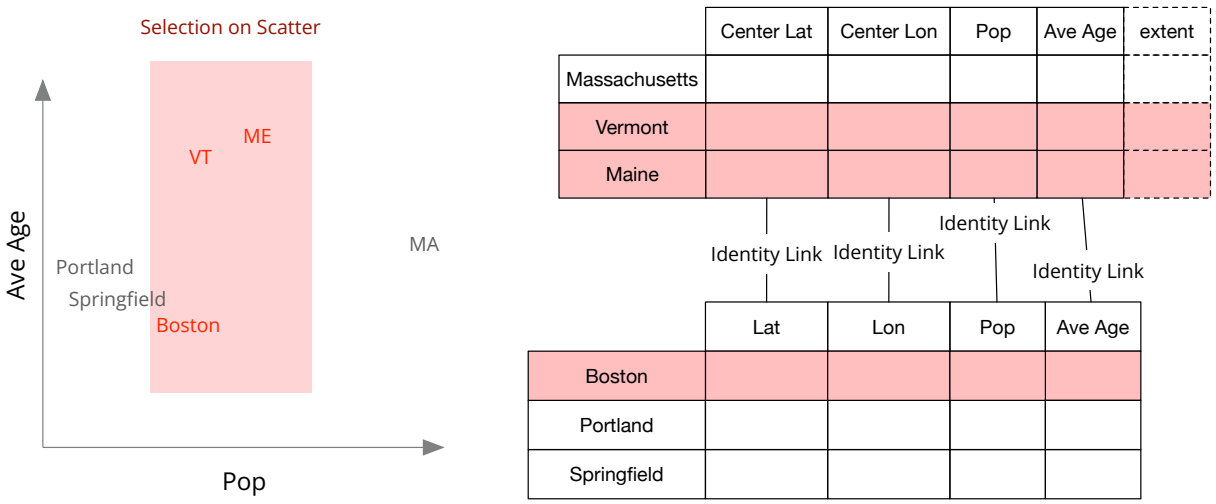
We use parent_components (representative points) to join Lat and Lon between cities and states. Rectangular selections on Lat/Lon do the expected thing (selecting both cities and states that lie within the rectangle) but selections of States on a table (or clicking to select individual states on a map) do NOT create a subset over cities. In this case we probably want to hide the extent component in the linking UI.



Note that in this linking case, we cannot plot states and cities together on a 2D scatter plot of Population vs. Ave Age (since they are not linked on those components). A 2D selection on a state scatter plot will never create a subset on cities. The user might be confused because they WANT to see the subset of cities that corresponds to the subset of states selected.

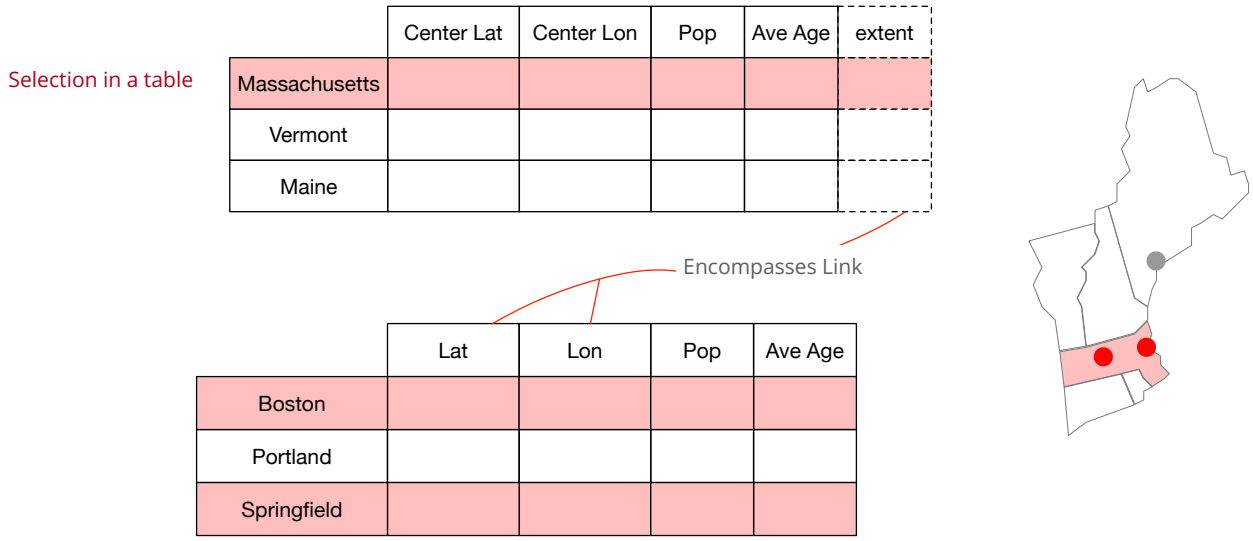


If Pop and Ave Age ARE linked, then the user can plot cities and states on the same 2D Scatter Viewer. In this case it's quite clear that the user is defining a subset over Pop and Average Age and probably wants to get the states and cities that individually fall within this selection and NOT the cities that also fall within the selection of states.

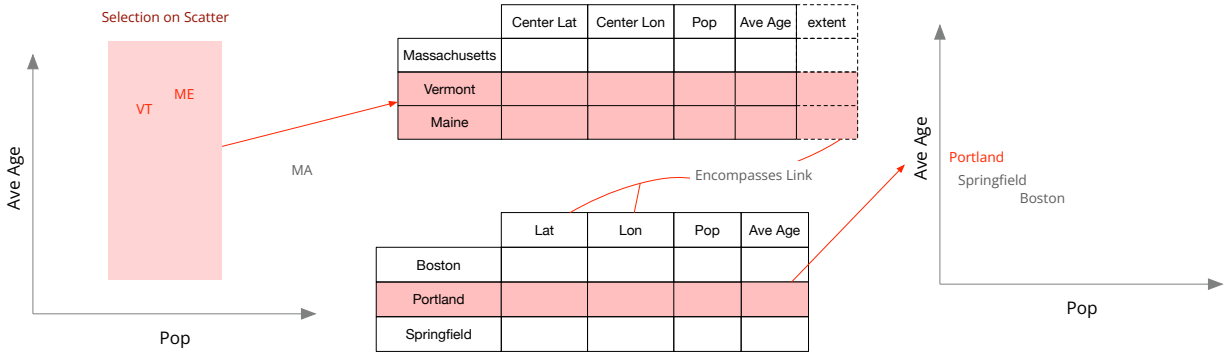


Case B:

We use an EncompassesLink to directly join the extent of the states to the Lat/Lon of the cities. Now a table or a map click selects the cities included in the selected states (behind the scenes, glue is translating the SubsetState from an ElementSubsetState to RoiSubsetState). In this case, we need to provide clear guidance in the linking UI about what is being done since the user will probably think they should join on Lat/Lon unless those parent_columns are hidden or shown in a different way that makes it clear they are special components.

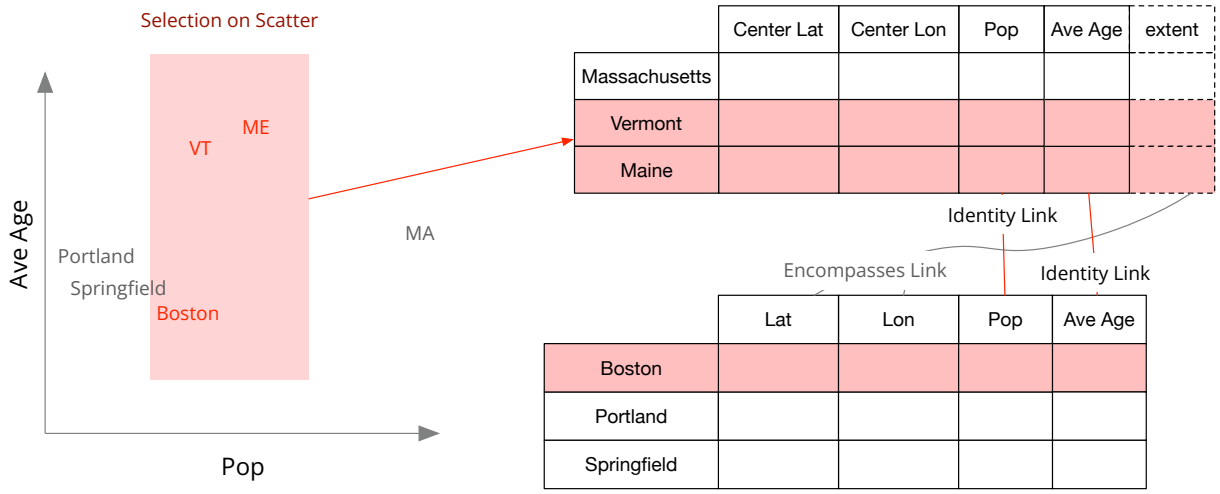


In this case, if we make a selection in a 2D scatter plot of Pop vs. Ave Age, we should get the subset of cities within those states. In the case where we're viewing the same components on cities and states that's a little counter-intuitive, but in the case where we're viewing totally different components on cities it allows us to select interesting subsets of states and see where those subsets of cities lie.



In the special case where we have made a selection on components that are ALREADY linked, regular glue identity links should probably take precedence over EncompassesLinks. Consider the situation shown below.

If we allow this subset to propagate through the EncompassesLink then selections on the scatter plot will mysteriously highlight other points (i.e. when you select a region that contains



The Identity Links have to override the Encompasses Links so that ROIs on a scatter plot to NOT suddenly highlight extra points on the scatter plot

Maine but not Portland, Portland will be selected anyway because it is Encompassed within Maine). We should first check to see if subsets can propagate through anything other than EncompassesLink and only if they cannot should we allow them to propagate through EncompassesLinks.

Note that all of the above discussion could also work in reverse. That is, instead of (or in addition to) an Encompasses Link, we could have a SelectsRegion type link. This would be useful in the cases of a one-to-one correspondence between regions and points (e.g. state capitols) so that selections over state capitols could select states in another viewer. The correspondence would not HAVE to be one-to-one. In the case of genomics, there are regions-of-interest called TADs that contain many genes. A biologist might be interested in identifying the subset of TADs that contain at least one gene in a subset of interesting genes.

Explicit Conversion

By contrast, requiring the user to explicitly request a conversion between a selected set of regions and the subset state representing the union of their extent offloads some of the complexity to the user. Behind the scenes this is a request to translate a SubsetState from whatever kind of SubsetState it was defined as into the RoiSubsetState that covers the same points in the region-definition dataset, but is now an ROI over the union of their extent.

We experimented with this in one version of glue genes, building a toolbar item in the table viewer that could be used to translate a subset of selected regions-of-interest into a the union of their extent. This was a suboptimal workflow, as the user had to come back to a specific viewer to update the definition of the subset.

It might be easier to put a context menu + menubar item that allowed the user to “translate subset to include all points in the region”. The context menu in the data collection area is not very discoverable however.

Another option that combines the user's explicit request to translate a subset to a region with the ability to seamlessly update that translation would be to allow the user (through a Viewer toolbar or menu item) to request the translation but use some kind of HubListener to provide continuous translation of the underlying SubsetState as long as it doesn't change "too much" — this is tricky because if the user changes the the underlying SubsetState to rely on components that aren't present in the extended dataset then the translation will have to stop, otherwise the new subset will be null, which is not what the user asked for. We could just check for this and disable automatic translation if the translation ever fails (perhaps with a warning dialog) or we could change the label or icon of the GroupedSubset to reflect the fact that it is now auto-translating.

Related Code

We already have DaskComponent(Component), CategoricalComponent(Component) and DataTimeComponent(Component) to handle specific data-types and methods of access that are a bit different from simple numpy arrays.

We can look at GeoPandas for an example of how to handle extended information next to tabular information.