Basic Design Principles and Code Structure

This is a Shiny app, with standard Shiny components:

- **server.R**: server-side functionality
- **UI.R**: static user-interface components
- **global.R**: static code and data that can be loaded at spin-up of the server and shared between instances of the app

The app consists of a set of distinct functional components, shown in the user interface in individual bootstrap (BS) collapse panels. Code for each component is located in a separate R file. Components access the static datasets and helper functions defined in **global.R**. Communication between components is via the shared reactive datastrutures and associated functions defined in **SharedComponents.R**. Each component is responsible for maintaining its own internal datastructures and input and output widgets. At a minimum, each component must define a **renderUI** expression that renders the output provided by the component into an appropriate Shiny output as defined in **UI.R**. A component may also want to add widgets and/or additional output to the bottom section of the sidebar (via InfoPane functionality provided in **SharedComponents.R**), and it may want to add an observer of the form **observeEvent(input$UIPanels, { if (input$UIPanels == 'Name of this component') <do something> })**, e.g., to clear the InfoPane or sync it's data and UI output with the current selection and other shared data structures.
Note: Current components in the app need to use crosstalk and other mechanisms in addition to Shiny reactive data structures to maintain selection state and communicate that state to local widgets. Allowing these components to synchronize their state simultaneously can lead to a tug-of-war situation that hangs the app. To prevent this, components that use mechanisms other than Shiny reactive datastructures should avoid using reactive expressions that trigger automatic updates to their selection. Instead, such components should monitor **input$UIPanels**, and only propagate selection information when their own panel is open.

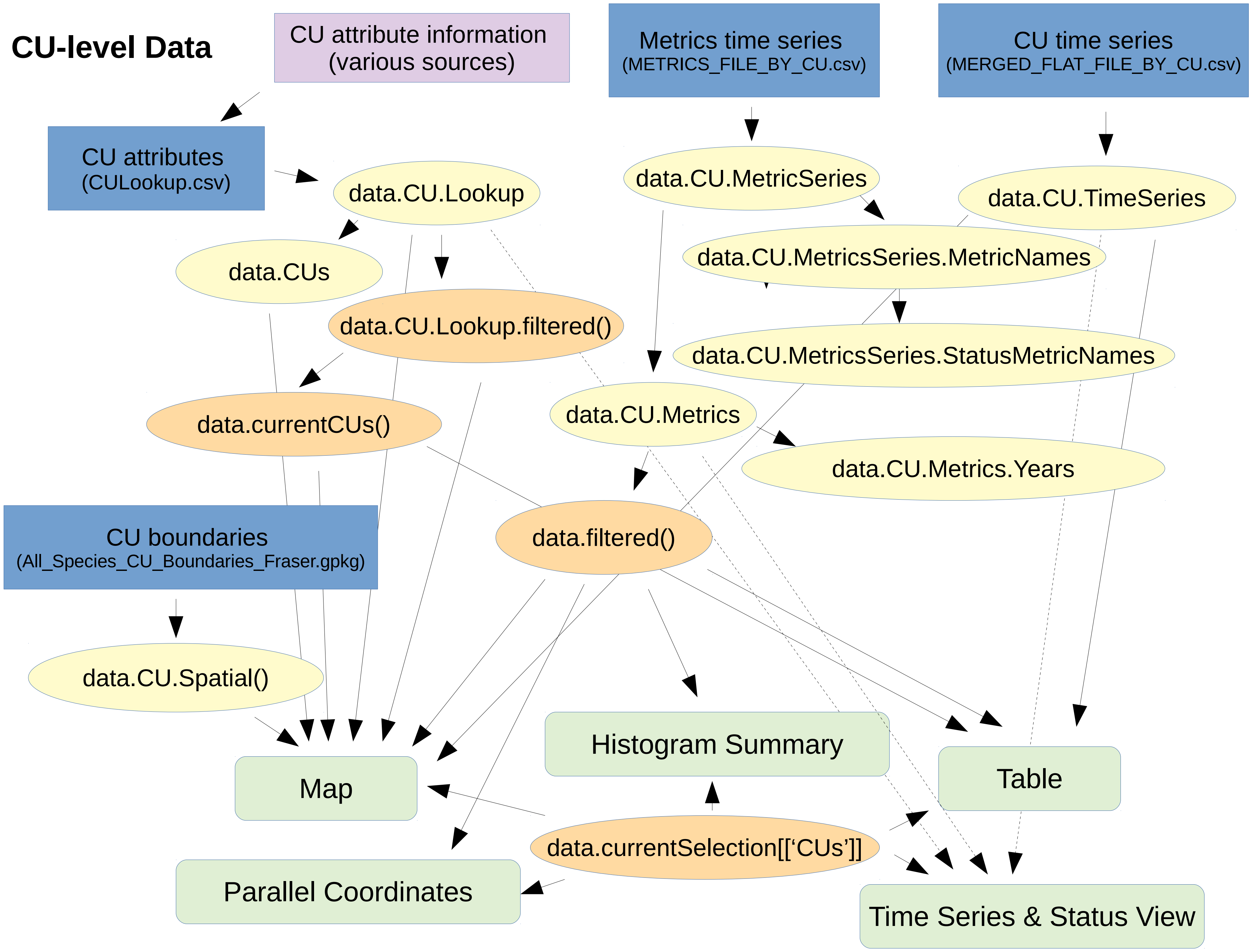
Existing components:

- **DataFilters**: Widgets for data filtering. Code in **DataFilters.R**
- **Map**: Interactive map. Code in **Map.R**
- **Parcoords**: Interactive parallel coordinates plot. Code in **Parcoords.R**
- **TSPlots:** Time series plots and status summary. Code in **TSPlots.R**
- **Table:** Interactive table and download. Code in **DataTable.R**
- **HistoSummary:** Histograms of metric values. Code in **HistogramSummary.R**

How to go about adding a new component::

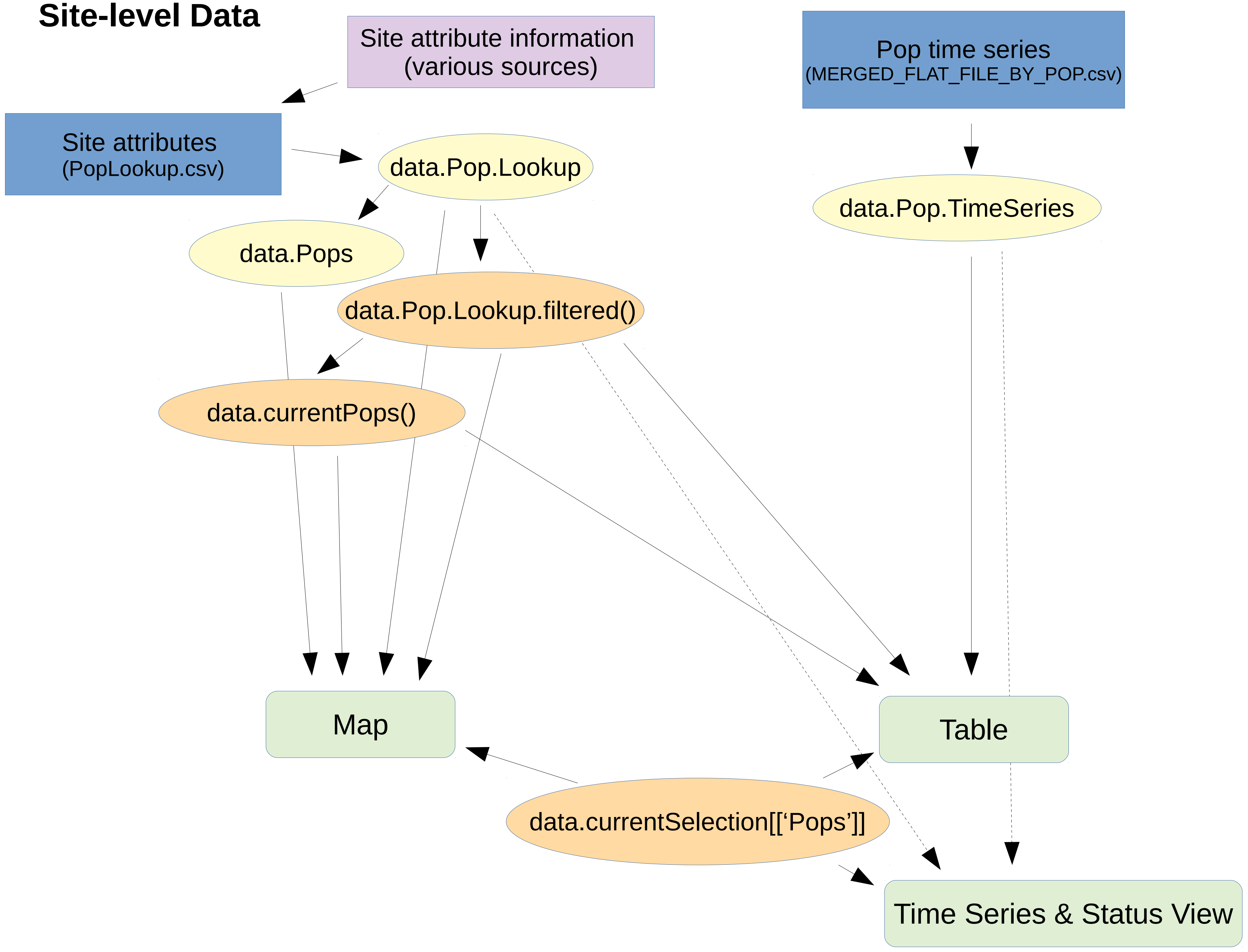
1) In **UI.R** add the line
   bsCollapsePanel(title = "your title", uiOutput("box_YourComponent"), value='YourComponent', ...)
2) Create **YourComponent.R**. Rely on what is defined in **global.R** and **SharedComponents.R** to get access to data and selection info.
3) At a minimum, **YourComponent.R** needs to provide a render statement of the form
   output$box_YourComponent <- renderUI({ ... your render code here ...})
4) Input and output widgets may also be added dynamically to the sidebar – use the InfoPane functions defined in **SharedComponents.R** for this.
5) In **server.R** add the line
   source('YourComponent.R', local=TRUE)
6) Since your code needs to be local to **server.R**, namespacing through putting the code into an R package is not an option. Preface names of any functions and reactive datastructures you define with an identifier uique to your component to avoid poluting the server namespace.

CU-level Data

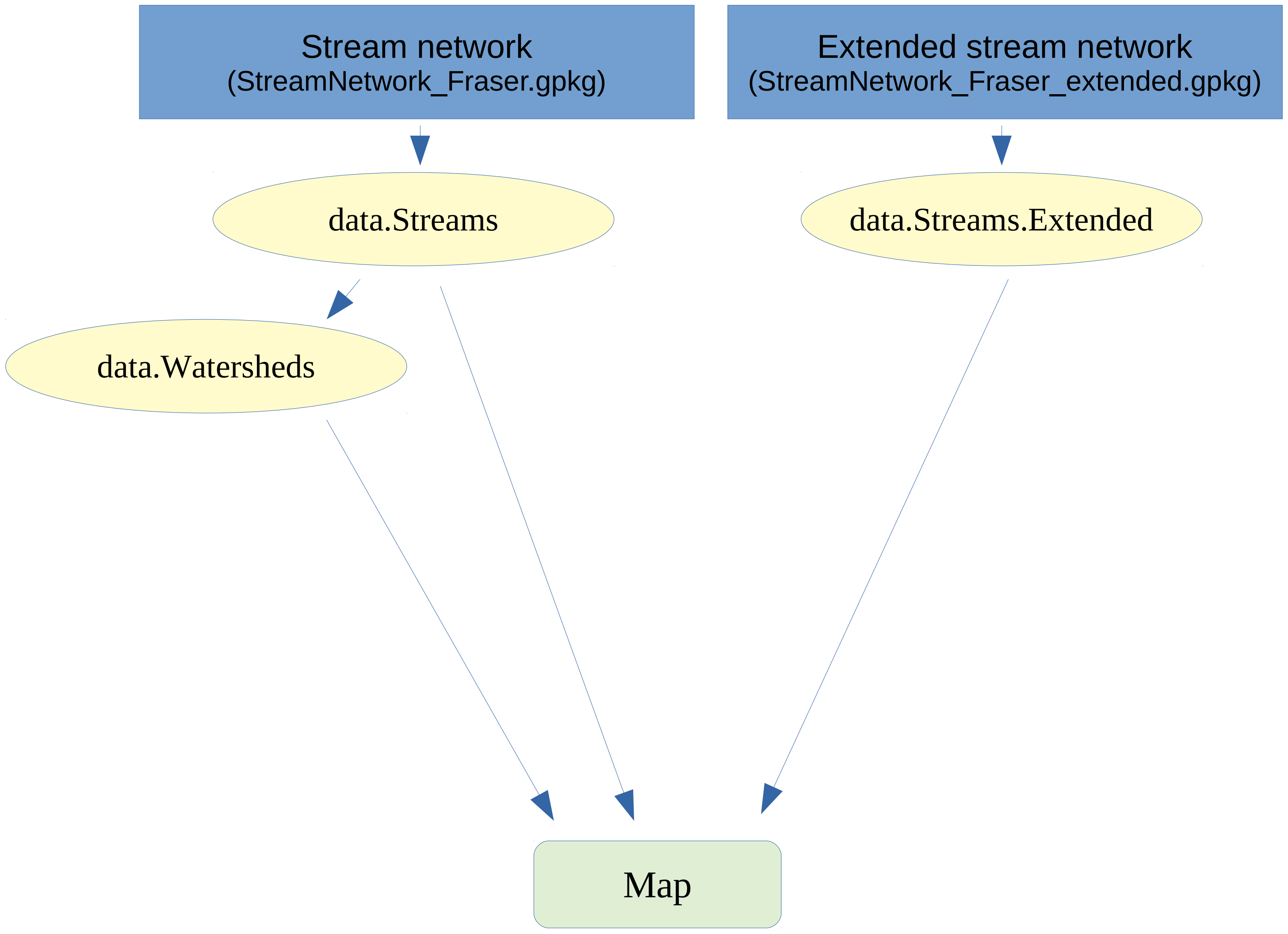


CU attribute information (various sources)
Metrics time series (METRICS_FILE_BY_CU.csv)
CU time series (MERGED_FLAT_FILE_BY_CU.csv)
CU attributes (CULookup.csv)
data.CU.Lookup
data.CU.MetricSeries
data.CU.TimeSeries
data.CUs
data.CU.MetricsSeries.MetricNames
data.CU.Lookup.filtered()
data.CU.MetricsSeries.StatusMetricNames
data.currentCUs()
data.CU.Metrics
data.CU.Metrics.Years
CU boundaries (All_Species_CU_Boundaries_Fraser.gpkg)
data.filtered()
data.CU.Spatial()
Histogram Summary
Map
Table
data.currentSelection[['CUs']]
Parallel Coordinates
Time Series & Status View

Site-level Data


Site attribute information (various sources)
Pop time series (MERGED_FLAT_FILE_BY_POP.csv)
Site attributes (PopLookup.csv)
data.Pop.Lookup
data.Pop.TimeSeries
data.Pops
data.Pop.Lookup.filtered()
data.currentPops()
Map
Table
data.currentSelection[['Pops']]
Time Series & Status View

Stream Data


Stream network
(StreamNetwork_Fraser.gpkg)
Extended stream network
(StreamNetwork_Fraser_extended.gpkg)
data.Streams
data.Streams.Extended
data.Watersheds
Map

Components of the data flow diagrams:

An input file

A global data structure
(created at startup of shiny server
and available to all modules)

A reactive shared data structure
available to all modules
(filtered datasets and current selection)

A module
(functional component in the app)

Input Files (located in data directory)

From SoS database:

- **METRICS_FILE_BY_CU.csv**: CU metrics data
- **MERGED_FLAT_FILE_BY_CU.csv**: CU time series data
- **MERGED_FLAT_FILE_BY_POP.csv**: Pop time series data

Spatial data:

- **All_Species_CU_Boundaries_Fraser.gpkg**: Cu boundary polygons
- **StreamNetwork_Fraser.gpkg**: Stream segments with CU and site info
- **StreamNetwork_Fraser_extended.gpkg**: segments extended upstream

Lookup files:

- **CULookup.csv**: matching of CU_IDs across input files & CU attributes
- **PopLookup.csv:** matching of Pop IDs across input files & site attributes

Global Data Structures (defined in global.R)

- **data.CU.Lookup**: CU attribute table
- **data.CU.MetricsSeries**: contents of METRICS_FILE_BY_CU.csv
- **data.CU.Metrics**: data.CU.MetricsSeries rearranged, one col per metric
- **data.CU.MetricsSeries.Names**: names of the available metrics
- **data.CU.Metrics.Years**: available assessment years
- **data.CU.TimeSeries**: CU time series data
- **data.CUs**: CUs in dataset
- **data.Pop.Lookup**: site attribute table
- **data.Pop.TimeSeries**: site time series data
- **data.Pops**: sites in dataset

Reactive Shared Data Structures (defined in SharedComponents.R)

- **data.CU.Lookup.filtered()**: CU attribute table, all filters applied
- **data.filtered()**: CU metrics table, all filters applied
- **data.currentCUs()**: list of CUs currently in the filtered data
- **data.currentPops()**: list of sites currently in the filtered data
- **data.current.Selection()**: list of lists:
    - data.currentSelection[[‘CUs’]] holds list of selected CUs
    - data.currentSelection[[‘Pops’]] holds list of selected sites

Modules (bsCollapse panels)

- **DataFilters**: (code in DataFilters.R) widgets for data filtering
- **Map**: (code in Map.R) interactive map
- **Parcoords**: (code in Parcoords.R) interactive parallel coordinates plot
- **TSPlots:** (code in TSPlots.R) time series plots and status summary
- **Table:** (code in DataTable.R) interactive table and download
- **HistoSummary:** (code in HistogramSummary.R) histograms of metric values