**MULTIPLE DETAIL PANEL INVESTIGATION**

**SUMMARY PANEL INTERACTIONS WITH DETAIL**

1. **Loading**: Initial detail panel drawn from top left summary panel.

2. **Click Navigation**: Summary clicks move detail position

3. **Zoom**: Scroll zoom on summary affects detail display

4. **Dendrogram**: Summary Dendrogram click changes detail position and switches detail to ribbon mode

5. **Click-drag**: Summary click-drag changes/drags detail position

6. **Click-shift-drag**: Summary shift-click-drag changes detail side to sub-ribbon view based upon area selected

**DETAIL PANEL INTERACTIONS WITH SUMMARY**

1. **Click Navigation**: Mouse drag moves selection box on summary panel

2. **Zoom**: Zoom changes selection box in summary panel

3. **Label Selection**: Label clicks set selection marks on summary panel

4. **Search**: Search sets selection marks on summary panel

5. **Dendrogram**: Dendro selection sets selection marks on summary panel

6. **Mode Change**: Changing modes (std/ribbon) changes selection box on summary panel

7. **Click-drag-select**: sets selection marks on summary panel

**SUMMARY/DETAIL CODE REFERENCES**

**SUMMARY-SIDE VARs USED ON THE DETAIL-SIDE JS**

* **NgChm.SUM.mouseEventActive** - Mouse active var used in Detail
* **NgChm.SUM.BYTE\_PER\_RGBA** - Summary var used in detailHeatMap function
* **NgChm.SUM.minValues** - Summary var used in detail

**SUMMARY-SIDE FUNCTIONS CALLED FROM DETAIL-SIDE JS**

* **NgChm.SUM.drawSelectionMarks** - Called on selectDrag, detailInit, labelDrag, labelClick on detail screen
* **NgChm.SUM.drawTopItems** - Called on selectDrag, detailInit, labelDrag, labelClick on detail screen
* **NgChm.SUM.clearSelectionMarks** - Called on labelClick
* **NgChm.SUM.buildScatterBarPlotMatrix** - Called on detail drawScatterPlotColClassBar
* **NgChm.SUM.buildScatterBarPlotMatrix** - Called on detail drawScatterPlotRowClassBar
* **NgChm.SUM.setLegendDivElement** - Called for high/low/mid div elements in detail drawScatterPlotColClassBar
* **NgChm.SUM.drawColClassBarLegends** - Called when drawing detail class bar legend
* **NgChm.DET.detailDrawColClassBarLabels** - Draws ... on summary class bars when bar is not shown (references summary canvas and summaryChm).
* **NgChm.SUM.drawRowClassBarLegends** - Called when drawing detail class bar legend
* **NgChm.DET.detailDrawRowClassBarLabels** - Draws ... on summary class bars when bar is not shown (references summary canvas and summaryChm).
* **NgChm.SUM.webGlGetContext** - Used to get detail GL Context

**DETAIL-SIDE VARs USED ON THE SUMMARY-SIDE JS**

* **NgChm.DET.minLabelSize** - Used when setting minimum font size in **NgChm.SUM.setBrowserMinFontSize** – Checks detail labels to set minimum browser font size

**DETAIL-SIDE FUNCTIONS CALLED FROM SUMMARY-SIDE JS**

* **NgChm.DET.removeLabel** - Called on summaryInit to clear (...) covariate labels. Also used to remove labels when building summary covar canvas textures AND on summary redraw.
* **NgChm.DET.addLabelDiv** - Used to add (...)labels when building summary covar canvas textures.
* **NgChm.DET.setDetCanvasBoxSize** - Adjusts size of detail canvas box (used to draw selection) when the summary size is reset
* **NgChm.DET.setDetailDataSize** - Called from set sub ribbon view on summary side (summary drag-select).
* **NgChm.DET.setDetailDataHeight** - Called from set sub ribbon view on summary side (summary drag-select).
* **NgChm.DET.setDetailDataWidth** - Called from set sub ribbon view on summary side (summary drag-select).
* **NgChm.DET.getNearestBoxSize** - Called from set sub ribbon view on summary side (summary drag-select).
* **NgChm.DET.getNearestBoxHeight** - Called from set sub ribbon view on summary side (summary drag-select).
* **NgChm.DET.drawColClassBarLegend** - Called from detail side to draw the column covariate bar legend on detail side. (shared function with summary)
* **NgChm.DET.drawRowClassBarLegend** - Called from detail side to draw the row covariate bar legend on detail side. (shared function with summary)
* **NgChm.DET.labelElement.appendChild** - Called by summary setLegendDivElement for detail side. (shared function with summary)
* **NgChm.DET.getContigSearchRanges** - Called when drawing selection marks on summary side to get ranges to be marked.

**DETAIL-SIDE VARs CALLED FROM DENDROGRAM JS**

* **NgChm.DET.initialized** - Used to determine when to stop drawing dendrogram
* **NgChm.DET.mouseDown** - Used in the click dendrogram function

**DETAIL-SIDE FUNCTIONS CALLED FROM DENDROGRAM JS**

* **NgChm.DET.labelLastClicked** - Used in Dendrogram addSelectedBar to get starting/ending indices for clicked bars

**IDEAS FOR IMPLEMENTING MULTIPLE DETAIL PANELS**

* Have a Primary Detail Panel (PDP)
* The PDP would be the only detail panel tied directly to the Summary Panel
* Allow multiple Secondary Detail Panels (SDP) but these windows would not interact with the Summary Panel. (they are open to a limitation on the number but don’t see why it couldn’t be unlimited)
* SDPs would not affect the Summary panel EXCEPT to add boxes to show where the SDPs are located in the summary view.
* Each SDP would be loaded with the image of the PDP at the time of the request.
* While SDPs do not affect the Summary Panel, they want the ability to assign PDP status to any open DP.
* Viewer Search functionality would affect only the PDP
* Selections affecting Summary and resulting from search, only on the PDP
* SDPs would allow zoom and views (std and ribbon)
* Selections made on SDPs would only highlight labels on those panes
* Standard Linkouts would be possible from both the PDP and SDPs
* What do we do about scatterplot and pathways mapper? Should they be driven entirely off of the PDP? (I think so) Perhaps we do not make these menu items available in the SDP, keeping perhaps even not having a panel menu for these panels? (Bradley needs to think about this)
* If we allow scatterplot/pathways to interact with the SDP that spawned them, we probably don't want them to set the summary panel selections?
* PDF - changes to PDF process, perhaps as a minimum to have extra pages for each DP open.

DESIGN IDEAS

Create a DetailHeatMapManager JS file for dealing with multiple Detail pages.

The Manager will:

1. Hold an array of detail window data objects (one for each window)
2. Spawn a new detail window
   1. Assuming there was no previous window it will spawn one pointing at summary (1,1)
   2. If there is an active Detail window it will spawn it using the detail window object data