interactiveDisplay: A package for enabling interactive visualization of Bioconductor objects

Shawn Balcome

December 18, 2024

Contents

List of Figures

1 Introduction

interactiveDisplay makes use of Bioconductor visualization packages, the Shiny [8] web framework, the D3.js visualization JavaScript library, and other libraries to produce various web applications built around Bioconductor objects.

Four popular Bioconductor data objects are currently supported: *GRanges*, *GRangesList*, *ExpressionSet* and *RangedSummarizedExperiment*. In addition, the second April 2014 release version now includes a method for data frames and is imported by the AnnotationHub [1] package.

This vignette will provide some background and guidance on the use of the current supported methods.

2 Installation

The webpage for the release version of interactiveDisplay is available at:

`http://bioconductor.org/packages/release/bioc/html/interactiveDisplay.html`

And the development version is available at:

`http://bioconductor.org/packages/devel/bioc/html/interactiveDisplay.html`

It can also be installed within the R console using BiocManager::install().

```
if (!requireNamespace("BiocManager", quietly=TRUE))
    install.packages("BiocManager")
BiocManager::install("interactiveDisplay")
```

interactiveDisplay is actively maintained and it may be desirable to use the developer branch of the package. If the user is willing to install development versions of Bioconductor packages, this can be accomplished by toggling the setting and reinstalling packages with BiocManager::install(version = "devel"). The development branch is actively changed, so users should expect to encounter errors. Generally, development and release versions of packages should not be mixed.

```
if (!requireNamespace("BiocManager", quietly=TRUE))
    install.packages("BiocManager")
BiocManager::install(version = "devel")
BiocManager::install("interactiveDisplay")
```

Because interactiveDisplay depends on many different R packages, some of which are used in only one individual method, many dependencies are suggested and will only be required when a method is run for the first time. This helps with load time when other packages (currently AnnotationHub) import methods from interactiveDisplay. If a required package is missing when a method is run, a message with instructions on obtaining it will be output in the R console.

3 Support

- Send me an email directly at balc0022@umn.edu
- Search, subscribe or contact the bioconductor or bioc-devel mailing lists at `http://www.bioconductor.org/help/mailing-list/`

4 Citation

```
citation("interactiveDisplay")
```

```
## To cite package 'interactiveDisplay' in publications use:
##
##   Balcome S (2024). _interactiveDisplay: Package for enabling powerful
##   shiny web displays of Bioconductor objects_.
##   doi:10.18129/B9.bioc.interactiveDisplay
##   <https://doi.org/10.18129/B9.bioc.interactiveDisplay>, R package
##   version 1.45.1,
##   <https://bioconductor.org/packages/interactiveDisplay>.
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {interactiveDisplay: Package for enabling powerful shiny web displays of Bioconductor
## objects},
##     author = {Shawn Balcome},
##     year = {2024},
```

```
##     note = {R package version 1.45.1},
##     url = {https://bioconductor.org/packages/interactiveDisplay},
##     doi = {10.18129/B9.bioc.interactiveDisplay},
##   }
```

5 Bioconductor Object Methods

5.1 *GRanges* and *GRangesList*

5.1.1 Object Background - *GRanges*

The *GRanges* object is a standardized container for genomic location data used in many Bioconductor packages. It is built on S4 classes from the the infrastructure package IRanges.

First, load the libraries needed for the examples in this document and load the example *GRanges* data provided with the interactiveDisplay package.

```
options(width=80)
options(continue=" ")
suppressMessages(library(ggplot2))
suppressMessages(library(interactiveDisplay))
suppressMessages(library(Biobase))
suppressMessages(library(GenomicRanges))
```

```
data(mmgr)
mmgr
```

```
## GRanges object with 1000 ranges and 2 metadata columns:
##          seqnames              ranges strand |       tx_id            tx_name
##             <Rle>           <IRanges>  <Rle> | <character>        <character>
##      [1]     chr5   53267178-53278539      + |       27296 ENSMUST00000147148
##      [2]     chr2 170346991-170427845      - |       40171 ENSMUST00000172093
##      [3]     chr7     4472475-4474584      + |       28623 ENSMUST00000084338
##      [4]     chr9 123772185-123790939      - |       37251 ENSMUST00000109152
##      [5]     chr7 127588698-127603605      + |       28623 ENSMUST00000038824
##      ...      ...                 ...    ... .         ...                ...
##    [996]     chr5 129715558-129718290      + |       24633 ENSMUST00000148260
##    [997]     chr4 115915120-115939926      - |       27296 ENSMUST00000147148
##    [998]     chr4 116598492-116599164      + |       27296 ENSMUST00000148260
##    [999]     chr5 124552958-124560784      + |       37251 ENSMUST00000038824
##   [1000]     chr2 170482708-170497145      - |       40171 ENSMUST00000172093
##   -------
##   seqinfo: 9 sequences from an unspecified genome
```

Each row in the *GRanges* object represents an individual range using the IRanges Bioconductor object to store the first and last position on the sequence. The chromosome and strand designation are both stored using run-length encoding (rle) for efficiency. The sequence lengths of are also stored in the *GRanges*. Any columns to the right of the pipe symbols in the object are metadata columns. Practically any R data object can be stored in the metadata, it is left up to the user and is often dependent on the experimental context of the data that is being conformed to the *GRanges* standard. For instance, to represent interactions between ranges, it is possible to store a *GRanges* object within another *GRanges* object's metadata. In the example here, transcript information has been inserted in the metadata.

5.1.2 Object Background - *GRangesList*

The *GRangesList* class is a container for storing a list of *GRanges* objects, the class is instanciated with constructor function GRangesList().

```
grl <- GRangesList(list(mmgr,mmgr))
```

GRangesList objects are useful for compound objects, an example given in the documentation being spliced transcripts that are are made up of exons [4]

```
data(mmgrl)
mmgrl
```

```
## GRangesList object of length 3:
## $gr1
## GRanges object with 189 ranges and 2 metadata columns:
##         seqnames              ranges strand |       tx_id            tx_name
##            <Rle>           <IRanges>  <Rle> | <character>        <character>
##     [1]     chr1   67585053-67587036      + |       14535 ENSMUST00000038824
##     [2]     chr1 185321183-185329396      - |       14535 ENSMUST00000172093
##     [3]     chr1   86545392-86582547      - |       14526 ENSMUST00000147148
##     [4]     chr1 180945962-180948838      + |       52581 ENSMUST00000109152
##     [5]     chr1 191894074-191907475      - |       14535 ENSMUST00000148260
##     ...      ...                 ...    ... .         ...                ...
##   [185]     chr3   35918998-35930119      - |       14526 ENSMUST00000109152
##   [186]     chr3   30906985-30969405      - |       14535 ENSMUST00000167068
##   [187]     chr3 121155865-121171694      - |       52581 ENSMUST00000147148
##   [188]     chr3   29952782-29997702      - |       24633 ENSMUST00000033088
##   [189]     chr3   67430096-67476529      + |       28623 ENSMUST00000084338
##   -------
##   seqinfo: 6 sequences from an unspecified genome
##
## $gr2
## GRanges object with 205 ranges and 2 metadata columns:
##         seqnames            ranges strand |       tx_id            tx_name
##            <Rle>         <IRanges>  <Rle> | <character>        <character>
##     [1]     chr1 67585053-67587036      + |       14535 ENSMUST00000038824
##     [2]     chr1 86545392-86582547      - |       14526 ENSMUST00000147148
##     [3]     chr1 87264363-87310836      + |       27296 ENSMUST00000144433
##     [4]     chr1 88318913-88388851      + |       37251 ENSMUST00000109152
##     [5]     chr1 59516264-59634508      + |       40171 ENSMUST00000038824
##     ...      ...                ...    ... .         ...                ...
##   [201]     chr8 21576640-21577426      + |       24633 ENSMUST00000144433
##   [202]     chr8   9206001-9771018      - |       28623 ENSMUST00000167068
##   [203]     chr8 40862412-40922308      + |       52581 ENSMUST00000172093
##   [204]     chr8 26119476-26129546      + |       14526 ENSMUST00000147148
##   [205]     chr8 25518895-25555658      + |       14535 ENSMUST00000084338
##   -------
##   seqinfo: 6 sequences from an unspecified genome
##
## $gr3
## GRanges object with 115 ranges and 2 metadata columns:
##         seqnames            ranges strand |       tx_id            tx_name
##            <Rle>         <IRanges>  <Rle> | <character>        <character>
##     [1]     chr2 32606961-32620804      + |       27296 ENSMUST00000172093
##     [2]     chr2 73908731-73911282      - |       40171 ENSMUST00000167068
```

```
##     [3]     chr2 30392430-30395911      + |       24633 ENSMUST00000148260
##     [4]     chr2 32221970-32222054      + |       27296 ENSMUST00000172093
##     [5]     chr2 66440896-66511249      + |       14535 ENSMUST00000033088
##     ...      ...                ...    ... .         ...                ...
##   [111]     chr8 21576640-21577426      + |       24633 ENSMUST00000144433
##   [112]     chr8   9206001-9771018      - |       28623 ENSMUST00000167068
##   [113]     chr8 40862412-40922308      + |       52581 ENSMUST00000172093
##   [114]     chr8 26119476-26129546      + |       14526 ENSMUST00000147148
##   [115]     chr8 25518895-25555658      + |       14535 ENSMUST00000084338
##   -------
##   seqinfo: 6 sequences from an unspecified genome
```

5.1.3 Method

The display() function in interactiveDisplay will start a Shiny web application tailored to the particular supported object passed as an argument. Both the *GRanges* and *GRangesList* data objects can be visualized as follows:

```
display(mmgr)
```

```
display(mmgrl)
```

It is also important to note, the GRanges and GRangesList methods are designed to send a subset of the submitted object back to the console as a result when the GUI application is exited. If the user wants to store this new object, simply assign it as illustrated in the following examples.

```
new_mmgr <- display(mmgr)
```

```
new_mmgrl <- display(mmgrl)
```

While simple in concept, data cleaning and wrangling are universal to any workflow, so this is a reasonable first implementation of Shiny's capability of returning results back to R.

5.1.4 UI

Unlike other visualization packages that are run in the console and lend themselves to walkthroughs and examples using the knitr package, interactiveDisplay's reliance on Shiny moves many of its functional components outside of the R console. This section will be a walkthrough of the user interface controls and features of the applications produced from the *GRanges* and *GRangesList* methods. Besides a couple small exceptions, these two methods are functionally similar and their descriptions have been consolidated together.

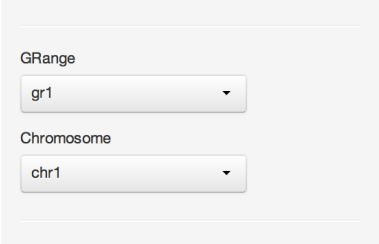


Figure 1: GRange, Chromosome drop-down menus

These dropdown UI elements are used to select a particular *GRanges* and ranges associated with a chosen chromosome (The *GRanges* dropdown is not present in the *GRanges* method). These ranges will be shown in the **Interactive Plot** and the subsetting operations in the other UI elements will apply to these active ranges.

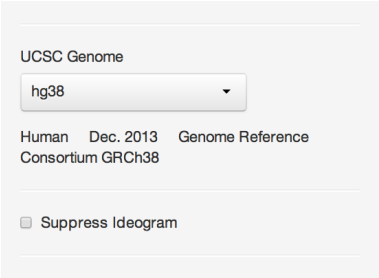


Figure 2: UCSC Genome drop-down, Suppress Ideogram checkbox

The **UCSC Genome** dropdown gives the user choices for the ideogram image shown above the trackplots. This selection is entirely determined by the user and the particular data in the GRange/GRangesList may not correspond to any of the available choices. This feature also depends on access to remote UCSC servers. Because the correct ideogram may not be available, it can be suppressed via the **Suppress Ideogram** checkbox element.

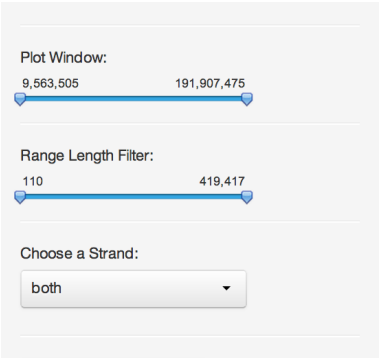


Figure 3: Plot Window Range slider, Length Filter slider, Choose a Strand drop-down

The **Plot Window** and **Range Length Filter** sliders and **Choose a Strand** dropdown provide the user with options for subsetting the original data to only the current ranges in view. The **Plot Window** is reflected by the highlighted region on the ideogram in addition to the trackplot below it. The **Range Length Filter** can be used to filter ranges shown by their respective lengths. The **Choose a Strand** dropdown gives the user a choice of displaying ranges located on positive, negative, or both strands.

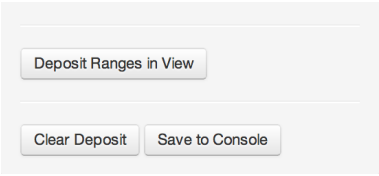


Figure 4: Deposit Ranges in View button, Clear Deposit button, Save to Console button

Besides subsetting which ranges are in the current view, the current subset of ranges can be saved. This is a two step process. Ranges of interest are first deposited, allowing the user to select different chromosomes/*GRanges* to manipulate and add to the list of deposited selections. Once the user is ready to return the data back to the console, **Save to Console** stops the Shiny session and returns the subsetted *GRanges* or *GRangesList* object. **Clear Deposit** resets all selections. The usefulness of this feature is left to the user's individual needs and is also very dependent on the data submitted. Upon inspection, ranges of interest may fall on the same chromosome or chromosome region, share specific metadata tags, or individual lengths. The Shiny UI offers an alternative to filtering away unwanted ranges iteratively in the console with multiple subsetting commands (and repeatedly printing output for validation), while providing instant visual feedback.

5.1.5 Plots

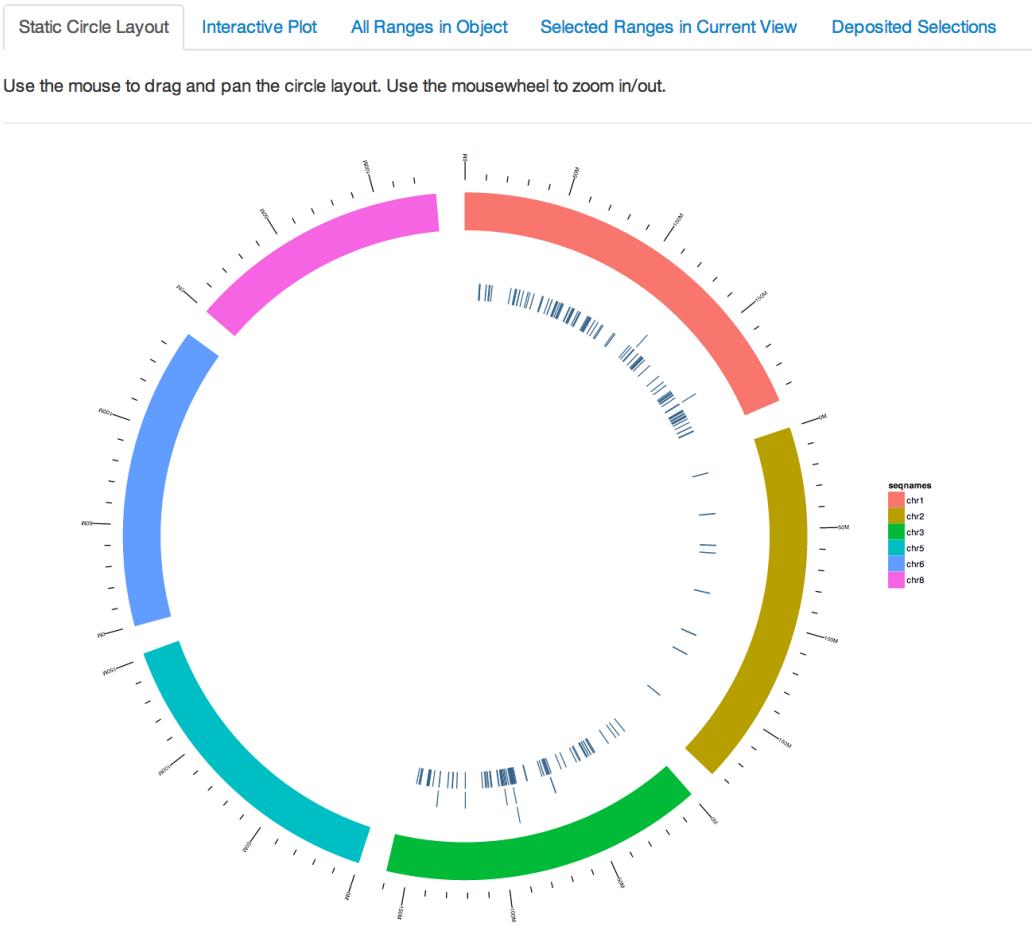


Figure 5: Static Circle Layout

The **Static Circle Layout** is a ggbio [11] based plot is provided for overview purposes and is not intended to represent the user's current subsetting operations. It provides the user with a broad summary of the data they submitted in its unaltered form.

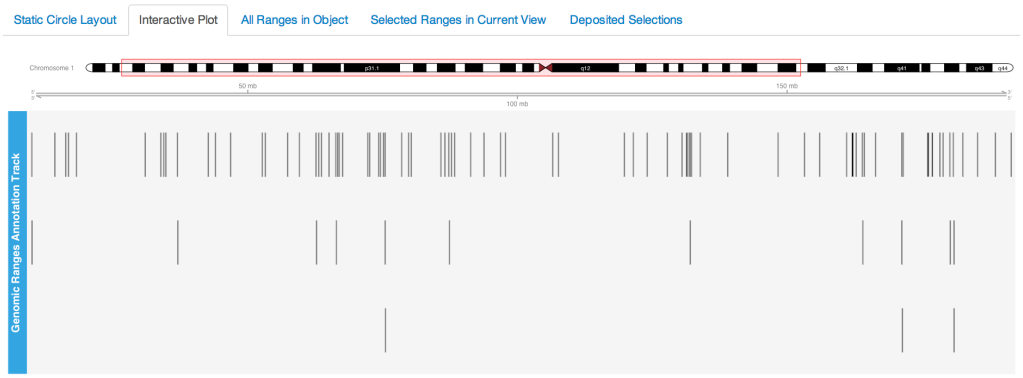
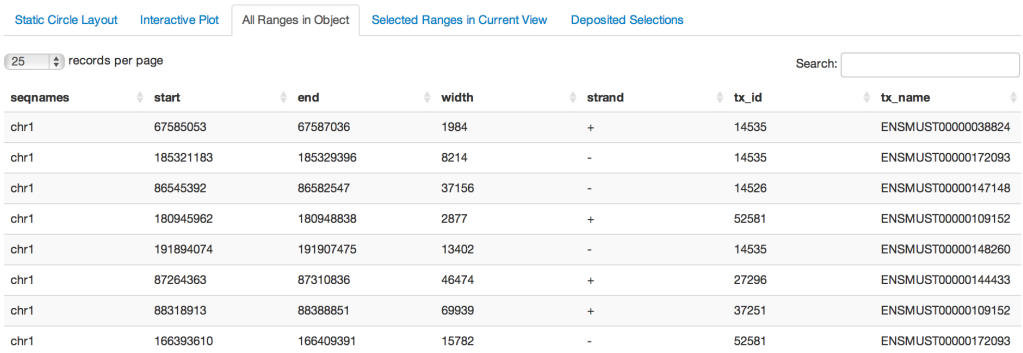


Figure 6: Interactive Plot

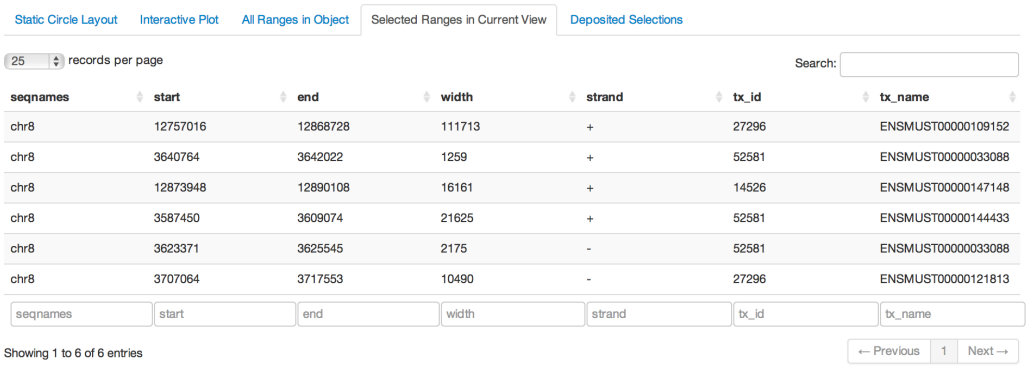
The **Interactive Plot** is the main view for these two methods. Trackplots are drawn with the Bioconductor package Gviz [3].



seqnames	start	end	width	strand	tx_id	tx_name
chr1	67585053	67587036	1984	+	14535	ENSMUST00000038824
chr1	185321183	185329396	8214	-	14535	ENSMUST00000172093
chr1	86545392	86582547	37156	-	14526	ENSMUST00000147148
chr1	180945962	180948838	2877	+	52581	ENSMUST00000109152
chr1	191894074	191907475	13402	-	14535	ENSMUST00000148260
chr1	87264363	87310836	46474	+	27296	ENSMUST00000144433
chr1	88318913	88388851	69939	+	37251	ENSMUST00000109152
chr1	166393610	166409391	15782	-	52581	ENSMUST00000172093

Figure 7: All Ranges in Object

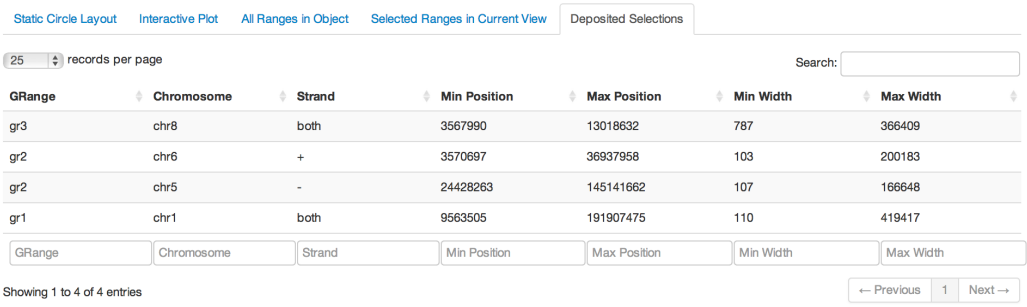
Using Shiny's advanced datatables, this is a searchable, sortable table of all ranges contained in the current *GRanges*. In the case of the *GRanges* method, this table contains all ranges in the object, and in the *GRangesList* method the table contains ranges in the current active *GRanges* selected from the dropdown.



seqnames	start	end	width	strand	tx_id	tx_name
chr8	12757016	12868728	111713	+	27296	ENSMUST00000109152
chr8	3640764	3642022	1259	+	52581	ENSMUST00000033088
chr8	12873948	12890108	16161	+	14526	ENSMUST00000147148
chr8	3587450	3609074	21625	+	52581	ENSMUST00000144433
chr8	3623371	3625545	2175	-	52581	ENSMUST00000033088
chr8	3707064	3717553	10490	-	27296	ENSMUST00000121813

Figure 8: Selected Ranges in Current View

This table is similar to the previous view, except it reflects the current state of subsetting operations performed by the UI and what is currently displayed in the trackplot.



GRange	Chromosome	Strand	Min Position	Max Position	Min Width	Max Width
gr3	chr8	both	3567990	13018632	787	366409
gr2	chr6	+	3570697	36937958	103	200183
gr2	chr5	-	24428263	145141662	107	166648
gr1	chr1	both	9563505	191907475	110	419417

Figure 9: Deposited Selections

This table view doesn't show individual ranges, instead it gives a summary of selected ranges that have been locked in with the **Deposit Ranges in View** button. Here the user can keep track of sets of ranges of interest that can be sent back to the console when the application is closed.

5.1.6 Metadata Tabset

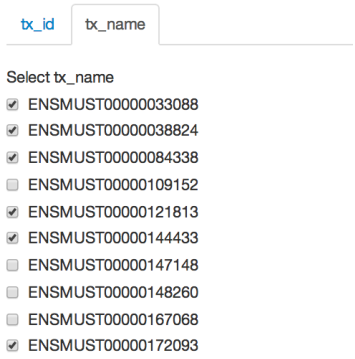


Figure 10: Dynamically Created Metadata Tabs

The lower tabset consists of individual tabs that are dynamically produced based on the metadata contents of the object submitted. Each tab has a list of checkboxes that allow the user to subset the selection based on the metadata. The subsetting is applied across all deposited selections. The usefulness of this panel really depends on the content of the object submitted. Objects containing no metadata columns obviously have no use for this UI, but metadata rich objects with categorical, biologically relevant labels can potentially be subset quickly according to them.

5.2 *ExpressionSet*

5.2.1 Object Background

The *ExpressionSet* class is a standardized container for gene expression data, specifically from a microarray.

```
data(expr)
expr
```

```
## ExpressionSet (storageMode: lockedEnvironment)
## assayData: 500 features, 26 samples
##   element names: exprs, se.exprs
## protocolData: none
## phenoData
##   sampleNames: A B ... Z (26 total)
##   varLabels: sex type score
##   varMetadata: labelDescription
## featureData: none
## experimentData: use 'experimentData(object)'
## Annotation: hgu95av2
```

We can view the expression values with the exprs() function.

```
exprs(expr)[1:10,1:7]
```

```
##                        A         B        C         D         E         F
## AFFX-MurIL2_at  192.7420  85.75330 176.7570 135.57500  64.49390  76.35690
## AFFX-MurIL10_at  97.1370 126.19600  77.9216  93.37130  24.39860  85.50880
## AFFX-MurIL4_at   45.8192   8.83135  33.0632  28.70720   5.94492  28.29250
## AFFX-MurFAS_at   22.5445   3.60093  14.6883  12.33970  36.86630  11.25680
## AFFX-BioB-5_at   96.7875  30.43800  46.1271  70.93190  56.17440  42.67560
## AFFX-BioB-M_at   89.0730  25.84610  57.2033  69.97660  49.58220  26.12620
## AFFX-BioB-3_at  265.9640 181.08000 164.9260 161.46900 236.97600 156.80300
```

```
## AFFX-BioC-5_at  110.1360  57.28890  67.3980  77.22070  41.34880  37.97800
## AFFX-BioC-3_at   43.0794  16.80060  37.6002  46.52720  22.24750  61.64010
## AFFX-BioDn-5_at  10.9187  16.17890  10.1495   9.73639  16.90280   5.33328
##                        G
## AFFX-MurIL2_at  160.5050
## AFFX-MurIL10_at  98.9086
## AFFX-MurIL4_at   30.9694
## AFFX-MurFAS_at   23.0034
## AFFX-BioB-5_at   86.5156
## AFFX-BioB-M_at   75.0083
## AFFX-BioB-3_at  211.2570
## AFFX-BioC-5_at  110.5510
## AFFX-BioC-3_at   33.6623
## AFFX-BioDn-5_at  25.1182
```

The structure of the *ExpressionSet* object [4]:

- expression data (assayData)
- sample metadata (phenoData)
- annotations and instrument metadata (featureData, annotation)
- sample protocol (protocolData)
- experimental design (experimentData)

5.2.2 Method

```
display(expr)
```

5.2.3 UI

The method for *ExpressionSet* has the most features of any of the current objects handled by interactiveDisplay. The main panel consists of three tabs that visualize the data as a heatmap, network and dendrogram. Gene ontology summaries of probes or selected probe clusters are available in tables in the lower tabset of the main panel.

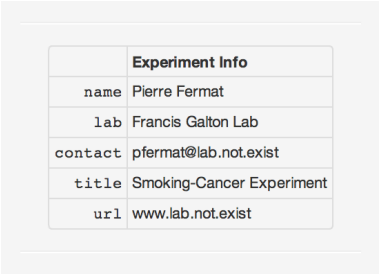


Figure 11: Experiment Info

The Experiment Info table at the top of the sidebar is entirely dependent on whether the *ExpressionSet* object submitted has these fields filled out.

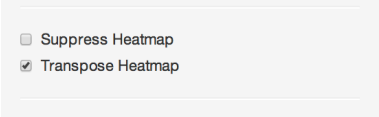


Figure 12: Suppress Heatmap checkbox, Transpose Heatmap checkbox

Depending on whether the parameter sflag=FALSE is passed to the the display() method and the selected dimensions of the *ExpressionSet*, replotting the heatmap can introduce delays. Using the checkbox to suppress the heatmap plotting can circumvent this. Also, depending on the relative dimensions of probes/samples, transposing the plot can make better use of the space available.

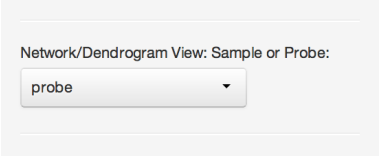


Figure 13: Network/Dendrogram View dropdown

This UI element simply toggles between displaying plots that represent distances/clustering across probes or samples.

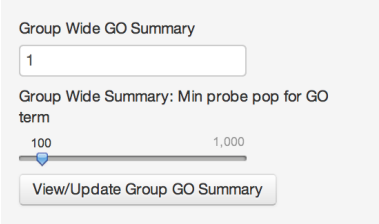


Figure 14: Group Wide GO Summary UI

This set of UI elements can be used to provide a gene ontology summary of a chosen cluster of probes in the network view. Simply mouseover a node within a cluster of interest and the cluster number will pop-up. Input that cluster number into the input field and hit the **View/Update GO Summary** button. This can take 10-30 seconds, which is why this is not automatically reactive. Adjusting the Minimum probe population for GO term allows the user to explore general or specific top ranking terms. The summary table is located in the second tab of the lower tabset.

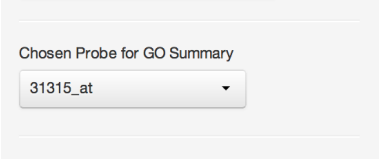


Figure 15: Individual Probe GO Summary dropdown

This drop-down contains options from the current subset of probes being visualized. Selecting a probe refreshes the first tab of the lower tabset with a GO summary for that probe, as well as associated gene names and Entrez IDs. When toggled to probe view, clicking on a probe node in the network view also refreshes this selection and may be more convenient.

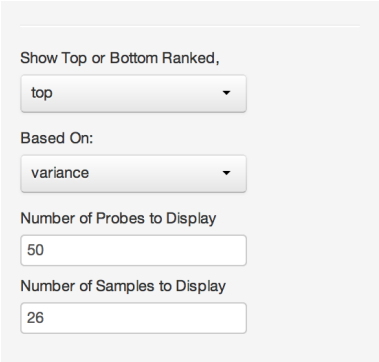


Figure 16: Subset UI

This UI set is used for subsetting the submitted *ExpressionSet* object. The initial values are set to the maximum number of samples and to a very conservative 20 probes. The user is free to expand out to the full dimensions.

Figure 17: Tweak Axis Label Font Size slider

Depending on the dimensions of the *ExpressionSet* object submitted/subset, it may be necessary to adjust label sizes to prevent label overlapping or illegibly small text.

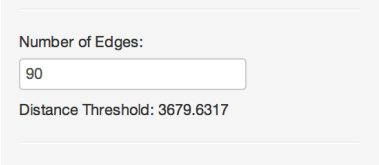


Figure 18: Edge/Distance UI

This UI element is used in the network view, allowing the user to determine the number of edges and the distance threshold represented in the graph.

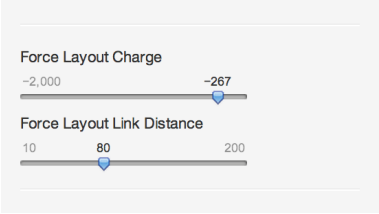


Figure 19: Force Layout sliders

This is a cosmetic control for the force directed graph, which allows the user to adjust how nodes spread themselves out and the length of their edges.

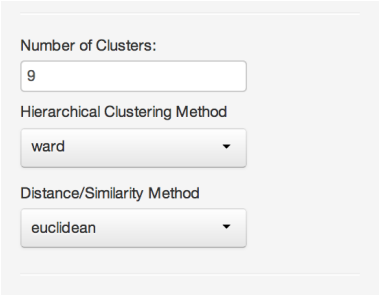


Figure 20: Clustering UI

These controls set the parameters for the distance and hierarchical clustering methods used. Only base R methods are currently implemented but more may be included in future releases as well as possible option of allowing the user to manually provide their own via a textbox element.

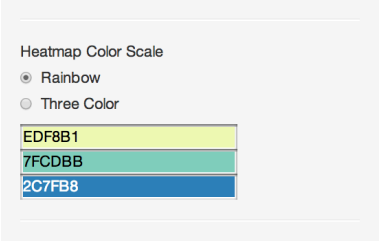


Figure 21: Color Picker UI

These elements affect the coloration of the heatmap. The color picker for the tricolor scale is not provided by the Shiny framework. This uses an additional JavaScript library which will be discussed in a later section. This is not just a cosmetic option, different modes can better visualize different data and accommodate color blind users.

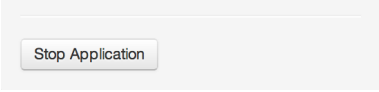


Figure 22: Stop Application button

This stops the application. The *ExpressionSet* method does not currently return any data back to the R console.

5.2.4 Plots

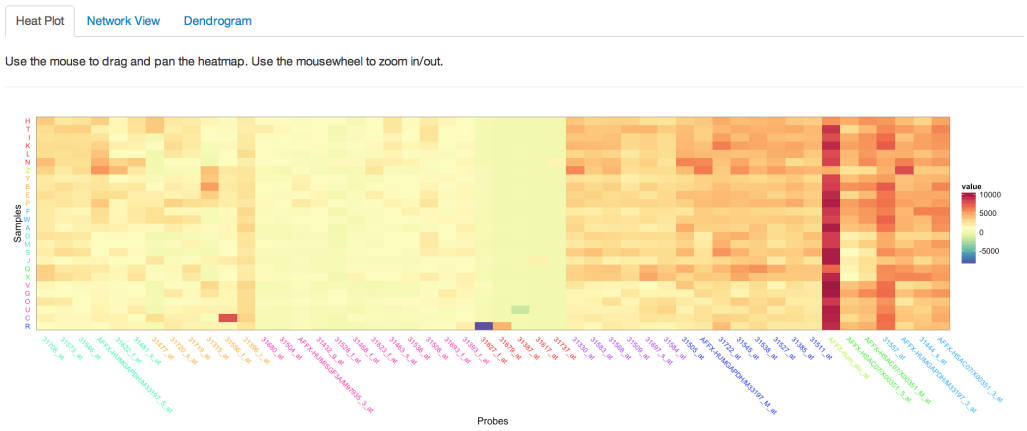


Figure 23: Heat Plot

Many packages on CRAN and Bioconductor have specialized heatmap functions, HeatPlus [6] has a specific function for *ExpressionSet* objects. Given that one of the goals of this package is modularity and reuse of code, the more generic plotting package ggplot2 [10] was used instead of other options. Often heatmap and dendrogram plots are paired together but given that there is a great deal of screen real estate taken up by tables, UI controls and on-screen documentation, the dendrograms are moved to a separate tab.

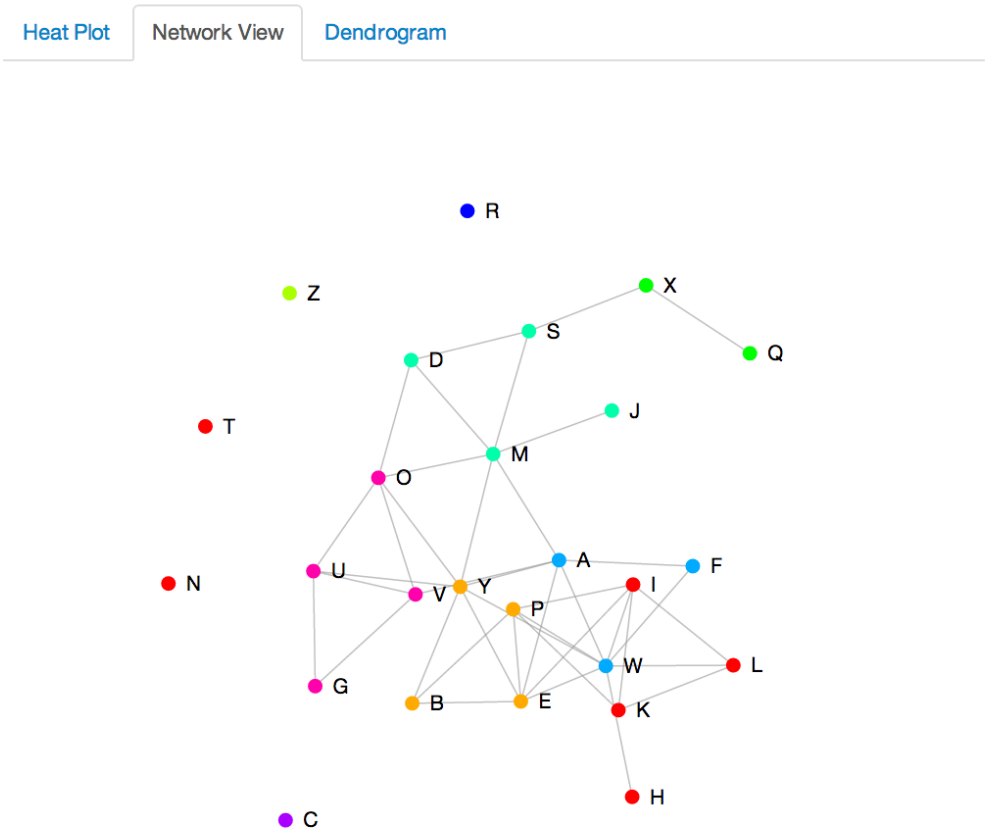


Figure 24: Network View - Samples

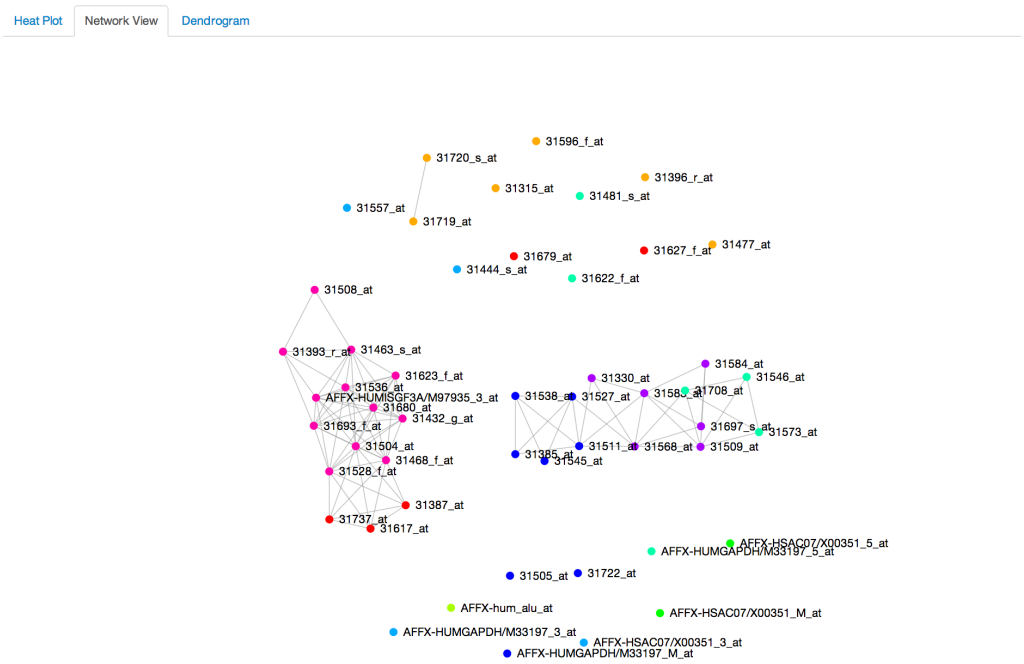


Figure 25: Network View - Probes

The **Network View** uses a D3.js based force directed graph to visualize distance/similarity between samples or probes based on user selected distance methods and threshold settings. Nodes in the graph are colored based on cluster settings. As stated previously, when the view is toggled to display probes, there is the added interface functionality of being to use the graph to conveniently gather GO information. When used in conjunction with the heatmap view, the network graph can quickly convey to the user which samples have similar profiles or which probes have similar expression across samples.

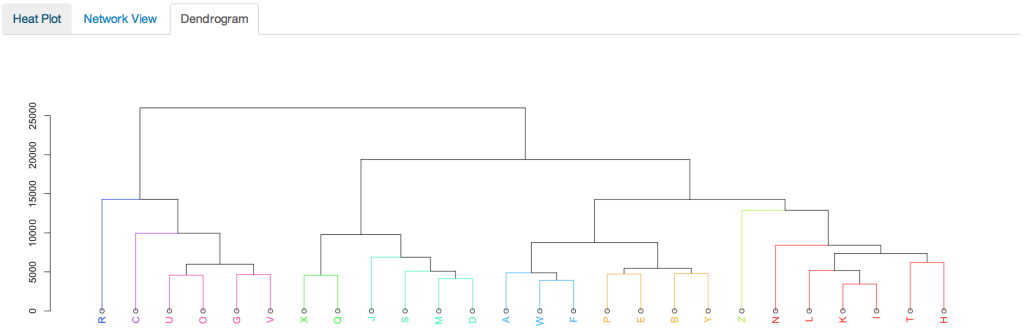


Figure 26: Dendrogram - Samples

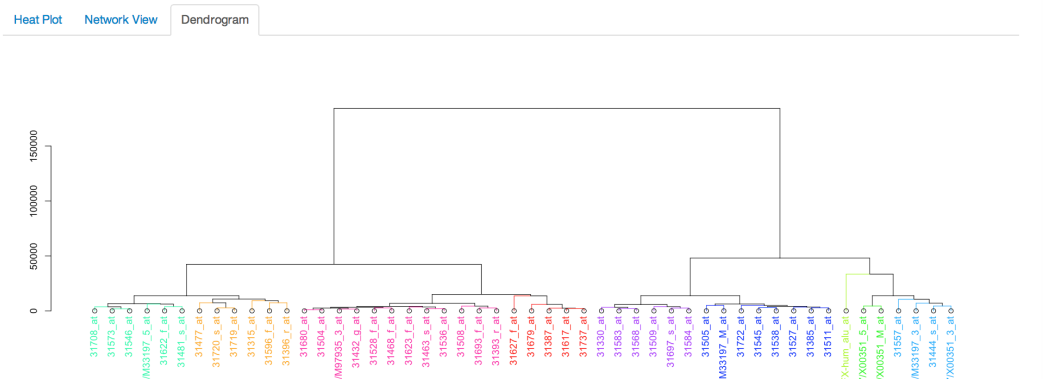


Figure 27: Dendrogram - Probes

The decision to separate the dendrogram plots from the heatmap was a difficult one, paired heatmap/dendrogram plots quickly convey clustering groupings alongside the data that they were generated from. However, due to the forementioned design/layout constraints, dendrograms are placed in their own tab. This keeps the already complex UI as clean as possible with adequate space for all plots. To accommodate this decision some care was taken to keep some consistency between views. The clustering colors are consistent across the three views and the sample/probe order is the same between the heatmap and dendrogram views. This consistency helps the user quickly switch between tabs and stay oriented with the data they are examining.

5.2.5 GO Tabset

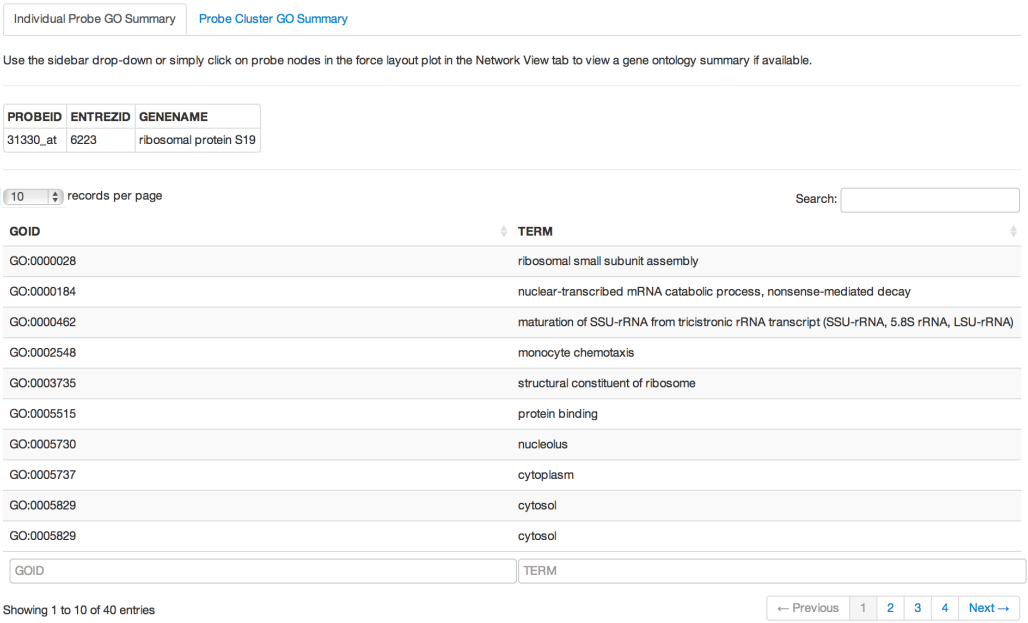
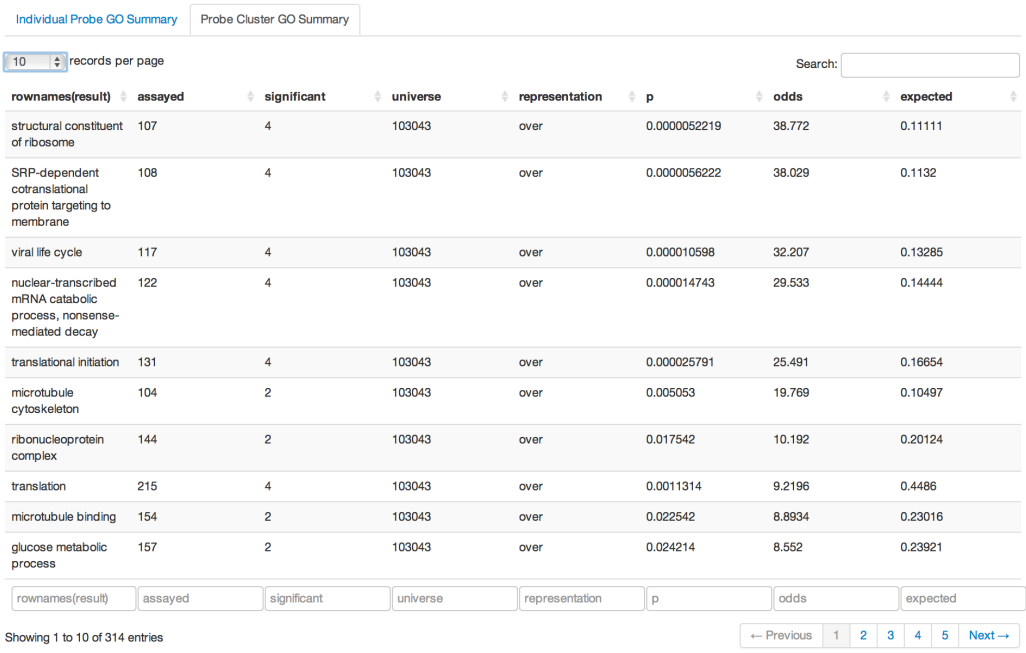


Figure 28: Individual Probe GO Summary

These summary tables provide Entrez ID and gene names for the selected probe and the top ranking results for GO descriptors. This table allows the user to manually choose and characterize probes that could be exhibiting differential expression in the submitted espressionSet.


Individual Probe GO Summary | Probe Cluster GO Summary

10 records per page — Search:

rownames(result)	assayed	significant	universe	representation	p	odds	expected
structural constituent of ribosome	107	4	103043	over	0.0000052219	38.772	0.11111
SRP-dependent cotranslational protein targeting to membrane	108	4	103043	over	0.0000056222	38.029	0.1132
viral life cycle	117	4	103043	over	0.000010598	32.207	0.13285
nuclear-transcribed mRNA catabolic process, nonsense-mediated decay	122	4	103043	over	0.000014743	29.533	0.14444
translational initiation	131	4	103043	over	0.000025791	25.491	0.16654
microtubule cytoskeleton	104	2	103043	over	0.005053	19.769	0.10497
ribonucleoprotein complex	144	2	103043	over	0.017542	10.192	0.20124
translation	215	4	103043	over	0.0011314	9.2196	0.4486
microtubule binding	154	2	103043	over	0.022542	8.8934	0.23016
glucose metabolic process	157	2	103043	over	0.024214	8.552	0.23921

Showing 1 to 10 of 314 entries — ← Previous 1 2 3 4 5 Next →

Figure 29: Probe Cluster GO Summary

This tab provides a table of the results of the hyperg() function from the GOstats package [2]. Ranked in order of p-value, the results give a general GO summary across the group of selected probes. Much like the previous table, this provides an interactive tool for characterizing differentially expressed sets of probes.

5.3 *RangedSummarizedExperiment*

5.3.1 Object Background

```
data(se)
se
```

```
## Loading required namespace:  SummarizedExperiment
```

```
## class: RangedSummarizedExperiment
## dim: 14599 4
## metadata(0):
## assays(1): ''
## rownames(14599): FBgn0000003 FBgn0000008 ... FBgn0261574 FBgn0261575
## rowData names(8): id baseMean ... pval padj
## colnames(4): untreated3 untreated4 treated2 treated3
## colData names(1): Condition
```

The *RangedSummarizedExperiment* class is similar to a dataframe where rows represent ranges (using *GRanges*/*GRangesList* and columns represent samples (with sample data summarized as a dataframe). It can contain one or more assays. Ranges have read counts associated with them [4].

5.3.2 Method

```
display(se)
```

5.3.3 UI

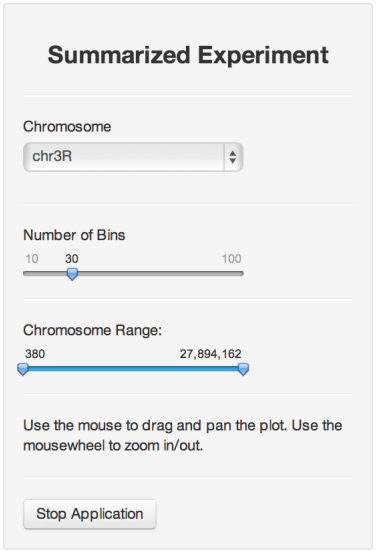


Figure 30: *RangedSummarizedExperiment* UI

The UI for the *RangedSummarizedExperiment* method is relatively simple. It is the best candidate of the existing methods for future development. The *RangedSummarizedExperiment* class is widely used, data rich object and has a lot of potential for improvement.

5.3.4 Plot

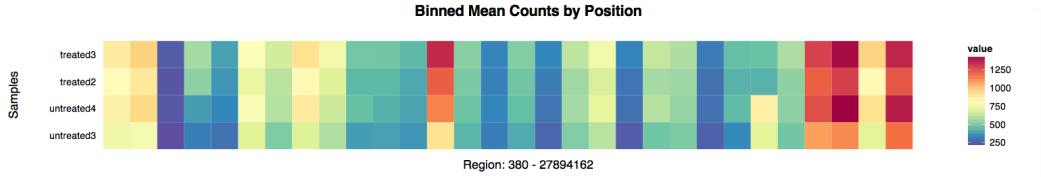


Figure 31: Binned Mean Counts by Position

Currently the plot gives a simple depiction of expression score/counts across regions of the selected chromosome. However, this plot currently obfuscates the relative coverage of the ranges of interest. It will likely need to be replaced with a multiple track plot in future releases.

6 Additional Functions/Methods

6.1 Dataframe - display()

```
display(mtcars)
```

Similar to the GRanges and GRangesList methods, the dataframe method can send a subset of the submitted object back to the console and can be stored in a new variable.

```
new_mtcars <- display(mtcars)
```

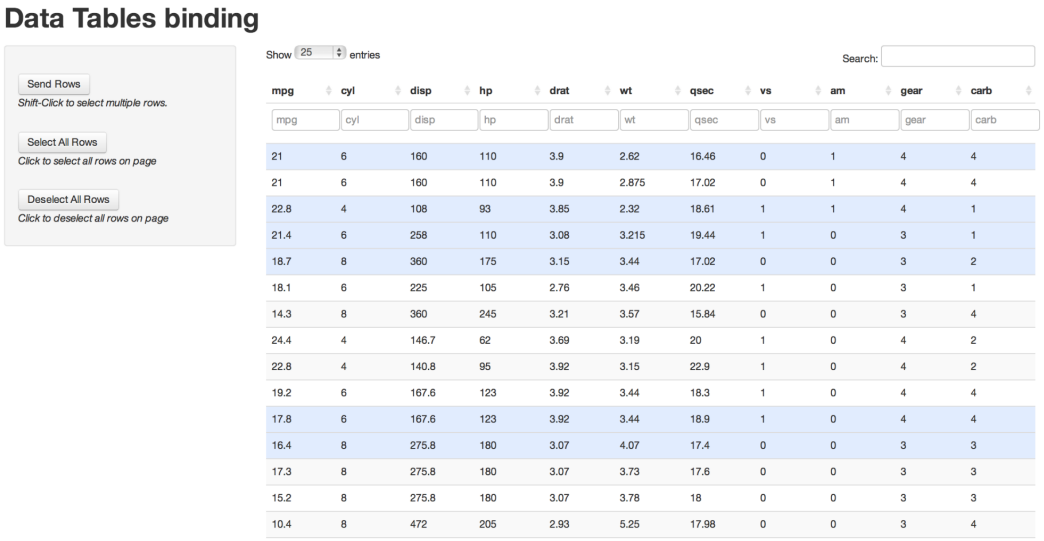


Figure 32: Dataframe Table

Despite the dataframe being a base R object and not exclusive to use with Bioconductor software, it's still useful to write methods for base R objects with genomic workflows in mind. The Bioconductor package AnnotationHub imports this dataframe method. This method makes use of Shiny's new advanced datatables with a slight modification to return selected rows back to the console.

6.2 Dataframe - simplenet()

```
simplenet(mtcars)
```

Figure 33: simplenet

Code to produce the force network tab from the *ExpressionSet* tab was recently converted to a stand-alone method for numeric dataframes in the developer branch of interactiveDisplay.

6.3 gridsvgjs()

One of the goals of this package is to provide users and future contributors with simple, modular functions that can be inserted into more complex methods or serve as the foundation or inspiration for future components. This motivation was behind the decision to export the helper function gridsvgjs(). This function takes any grid based plot, converts it to SVG format (see section on gridSVG), inserts JavaScript for the Pan/Zoom functionality (see JavaScript Libraries section) and initiates a local Shiny instance. The four display() methods make use of the majority of the code in gridsvgjs(). After considering the potential usefulness of doing this with any plot, I repackaged the code as a stand-alone function. The standard R graphics viewer can sometimes be restrictive, particularly for larger plots. Viewing a plot as a vector based rendering with pan and zoom capabilities seemed to be a desirable addition to the package. The inserted JavaScript functionality is relatively simple, but hopefully it will encourage more interactivity in the future.

```
data(mtcars)
qp <- qplot(mpg, data=mtcars, geom="density", fill=factor(cyl), alpha=I(.4))
plot(qp)
```

To view the plot in the browser with JavaScript Pan/Zoom capabilities:

```
gridsvgjs(qp)
```

7 Special Mention Components

7.1 Shiny

The Shiny web framework is the component that was the main inspiration behind the proposal for this package. The relative simplicity of prototyping applications and its use of existing universal web standards makes it an enticing piece of software to build an open source package around. It means, unlike some packages built on low level code, there is a possibility for other biologists to contribute to the open source development. Shiny provides the web service and predefined UI elements, but relies on existing R packages to provide the data analysis and plotting. The interactivity of Shiny apps depends on its reactive programming model.

`http://shiny.rstudio.com/`

7.2 Gviz/ggbio

Gviz and ggbio provide the biology specific plots in the *GRanges* and *GRangesList* methods in interactiveDisplay. Both offer track-based plotting methods for genomic data. Gviz was the initial choice for *GRanges*/*GRangesList* trackplots because it offered more generic, extensible plotting. Later in development, ggbio plotting was also added as it handles additional complex plotting methods.

7.3 gridSVG

Simon Potter's gridSVG [5] is an important asset in developing interactive R plots. His package converts a grid [7] based plot to SVG format directly. Two major plotting packages for R, lattice [9] and ggplot2 are built around grid (and many other packages depend on ggplot2), which gives gridSVG a lot of potential interaction with many Bioconductor libraries.

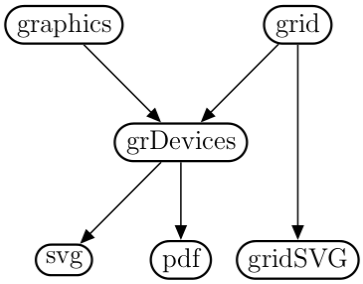


Figure 34: gridSVG [5]

8 JavaScript Libraries

8.1 Data-Driven Documents

Michael Bostock's JavaScript library, D3.js (`http://d3js.org/`), allows for highly customizable data visualizations using universal web standards. It can work along with Shiny to bind data to the Document Object Model (DOM). Unlike Shiny which handles the interactive elements but leaves R to handle the visualization, D3.js can handle both. Currently only the *ExpressionSet* method's Network View makes use of this library.

8.2 Zoom/Pan JavaScript libraries

This Zoom/Pan library is used in all four main display() methods and in gridsvgjs(). While a relatively simple feature, hopefully it will lead to more complex JavaScript interactivity that directly manipulates plots produced by R plotting packages. The Andrea Leofreddi's original JavaScript code was expanded upon by John Krauss to produce the version of this library used in interactiveDisplay.

Andrea Leofreddi

`https://code.google.com/p/svgpan/`

John Krauss

`https://github.com/talos/jquery-svgpan`

8.3 JavaScript Color Chooser

The colorpicker included in the *ExpressionSet* method should eventually be moved into the Shiny package itself. Many R users outside of the Bioconductor community use RColorBrewer and could make use of this UI element in their Shiny based applications.

Jan Odvarko

`http://jscolor.com/`

9 Acknowledgments

Shiny
Joe Cheng and Winston Chang
`http://www.rstudio.com/shiny/`

Force Layout
Jeff Allen
`https://github.com/trestletech/shiny-sandbox/tree/master/grn`

gridSVG
Simon Potter
http://sjp.co.nz/projects/gridsvg/

Zoom/Pan JavaScript libraries
John Krauss
https://github.com/talos/jquery-svgpan
Andrea Leofreddi
https://code.google.com/p/svgpan/

JavaScript Color Chooser
Jan Odvarko
http://jscolor.com/

Data-Driven Documents
Michael Bostock
http://d3js.org/

10 SessionInfo

```
sessionInfo()
```

```
## R Under development (unstable) (2024-11-20 r87352)
## Platform: aarch64-apple-darwin20
## Running under: macOS Ventura 13.7.1
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.5-arm64/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.5-arm64/Resources/lib/libRlapack.dylib;  LAPACK versi
##
## locale:
## [1] C/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## time zone: America/New_York
## tzcode source: internal
##
## attached base packages:
## [1] stats4    grid      stats     graphics  grDevices utils     datasets
## [8] methods   base
##
## other attached packages:
##  [1] GenomicRanges_1.59.1      GenomeInfoDb_1.43.2
##  [3] IRanges_2.41.2            S4Vectors_0.45.2
##  [5] Biobase_2.67.0            interactiveDisplay_1.45.1
##  [7] BiocGenerics_0.53.3       generics_0.1.3
##  [9] ggplot2_3.5.1             knitr_1.49
##
## loaded via a namespace (and not attached):
##  [1] SummarizedExperiment_1.37.0   KEGGREST_1.47.0
##  [3] gtable_0.3.6                  xfun_0.49
##  [5] lattice_0.22-6                vctrs_0.6.5
##  [7] tools_4.5.0                   tibble_3.2.1
##  [9] AnnotationDbi_1.69.0          RSQLite_2.3.9
## [11] Category_2.73.0               highr_0.11
## [13] blob_1.2.4                    pkgconfig_2.0.3
```

```
## [15] Matrix_1.7-1                 RColorBrewer_1.1-3
## [17] graph_1.85.0                 lifecycle_1.0.4
## [19] GenomeInfoDbData_1.2.13      compiler_4.5.0
## [21] stringr_1.5.1                Biostrings_2.75.3
## [23] munsell_0.5.1                httpuv_1.6.15
## [25] htmltools_0.5.8.1            interactiveDisplayBase_1.45.0
## [27] later_1.4.1                  pillar_1.10.0
## [29] crayon_1.5.3                 DelayedArray_0.33.3
## [31] cachem_1.1.0                 abind_1.4-8
## [33] mime_0.12                    genefilter_1.89.0
## [35] tidyselect_1.2.1             digest_0.6.37
## [37] stringi_1.8.4                dplyr_1.1.4
## [39] reshape2_1.4.4               splines_4.5.0
## [41] gridSVG_1.7-5                fastmap_1.2.0
## [43] SparseArray_1.7.2            colorspace_2.1-1
## [45] cli_3.6.3                    magrittr_2.0.3
## [47] S4Arrays_1.7.1               survival_3.8-3
## [49] RBGL_1.83.0                  XML_3.99-0.17
## [51] GSEABase_1.69.0              withr_3.0.2
## [53] scales_1.3.0                 UCSC.utils_1.3.0
## [55] promises_1.3.2               bit64_4.5.2
## [57] XVector_0.47.0               httr_1.4.7
## [59] matrixStats_1.4.1            bit_4.5.0.1
## [61] png_0.1-8                    memoise_2.0.1
## [63] shiny_1.10.0                 evaluate_1.0.1
## [65] rlang_1.1.4                  Rcpp_1.0.13-1
## [67] xtable_1.8-4                 glue_1.8.0
## [69] DBI_1.2.3                    annotate_1.85.0
## [71] jsonlite_1.8.9               R6_2.5.1
## [73] plyr_1.8.9                   MatrixGenerics_1.19.0
## [75] zlibbioc_1.53.0
```

11 References

[1] Marc Carlson and Sonali Arora. *AnnotationHub: A client for retrieving Bioconductor objects from AnnotationHub*. R package version 1.4.0.

[2] S Falcon and R Gentleman. Using GOstats to test gene lists for GO term association. *Bioinformatics*, 23(2):257–8, 2007.

[3] Florian Hahne, Steffen Durinck, Robert Ivanek, Arne Mueller, Steve Lianoglou, and Ge Tan. *Gviz: Plotting data and annotation information along genomic coordinates*. R package version 1.8.2.

[4] Michael Lawrence, Wolfgang Huber, Hervé Pagès, Patrick Aboyoun, Marc Carlson, Robert Gentleman, Martin Morgan, and Vincent Carey. Software for computing and annotating genomic ranges. *PLoS Computational Biology*, 9, 2013.

[5] Paul Murrell and Simon Potter. *gridSVG: Export grid graphics as SVG*, 2013. R package version 1.4-0.

[6] Alexander Ploner. *Heatplus: Heatmaps with row and/or column covariates and colored clusters*, 2014. R package version 2.11.0.

[7] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2013.

[8] RStudio and Inc. *shiny: Web Application Framework for R*, 2014. R package version 0.9.1.

[9] Deepayan Sarkar. *Lattice: Multivariate Data Visualization with R*. Springer, New York, 2008. ISBN 978-0-387-75968-5.

[10] Hadley Wickham. *ggplot2: elegant graphics for data analysis*. Springer New York, 2009.

[11] Tengfei Yin, Dianne Cook, and Michael Lawrence. ggbio: an r package for extending the grammar of graphics for genomic data. *Genome Biology*, 13(8):R77, 2012.