Extensions to the Model-View-Controller Package

Elizabeth Whalen

July 28, 2006

1 Overview

The *BioMVCClass* package extends the model and view classes that are defined in the *MVCClass* package. These new classes were not placed in the *MVCClass* package because we did not want that package to depend on the *Biobase*, *graph*, and *Rgraphviz* packages. The classes defined in the *MVCClass* and *BioMVCClass* packages let developers create model-view-controller applications of linked, interactive views of data. Please see the *MVCClass* Vignette for more information on that package. The *BioMVCClass* package depends on the *MVCClass*, *Biobase*, *graph*, and *Rgraphviz* packages.

2 Model Classes

The model class is responsible for storing and updating the data. These two functions are reflected in Figure 1, which shows the inheritance structure for the model classes. Here the virtual class, `gModel`, has the slots: modelData, linkData, virtualData, modelName, and modelVar. These five slots are common information that all models need. The modelData slot is the data set for this model, the linkData slot is a list of two functions, `toParent` and `fromParent`, which link this model to its parent and child models, respectively (see Section **??**), the virtualData slot is data pertaining to the views that needs to be stored with the model so that it can be shown in all views of this model, the modelName slot is the name of the model (and the name of the MVC), and the modelVar slot is a list of named model variables (for example, a statistic for each element in the model). Note that in Figure 1, the `gModel` class is defined in the *MVCClass* package and the `graphModel`, `exprModel`, and `gseModel` classes are defined in the *BioMVCClass* package.

For specific model classes, there are `graphModel`, `exprModel`, and `gseModel`. The `graphModel` class represents a model where the modelData has class `graph`, the `exprModel` class represents a model where the modelData has class `ExpressionSet`, and the `gseModel` class represents a model where the modelData has class `GSE`. The `GSE` class, which is defined in the *BioMVCClass* package, is discussed below in Section 2.1.

2.1 Gene Set Enrichment (GSE) Class

One type of data that we are interested in studying is data from a gene set enrichment (GSE) analysis. This data is bound in a class called GSE, which has the slots incidMat, gTestStat, gsTestStat, expData, and descr. The incidMat slot holds the incidence matrix that shows which genes belong to which gene sets, the gTestStat slot holds the test statistic that relates the gene to the phenotype, the gsTestStat slot

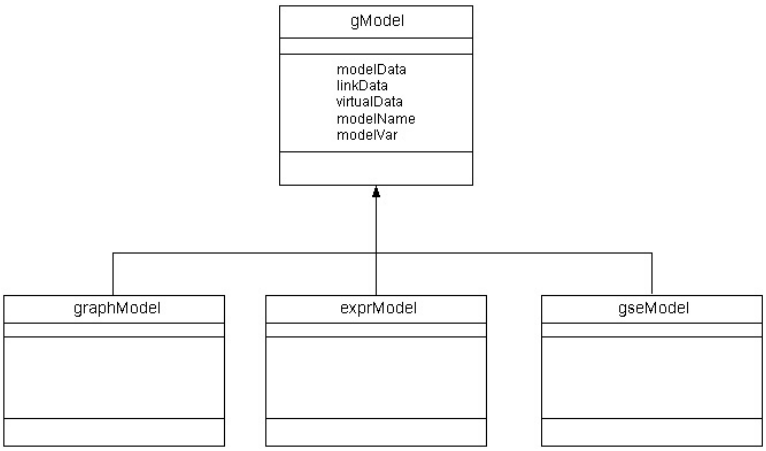

Figure 1: Inheritance for Model Objects.

holds the gene set test statistic that is calculated by multiplying the incidence matrix and the gene test statistics, the expData slot holds the experimental data, and the descr slot is a description of the gene sets being studied. The class definition for `GSE` is shown in Figure 2.

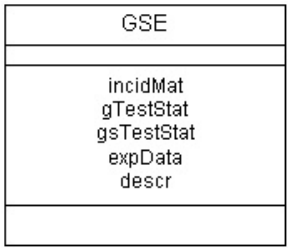

Figure 2: The Gene Set Enrichment (GSE) Class.

The `GSE` class is then used in the modelData slot of the `gseModel` class.

3 View Classes

The view classes represent the visual depictions of the model. All views need to store some common information and respond to certain events through methods. This consideration led to an object model where the different view classes inherit from a view virtual class, called `genView`. The object model for the view classes is shown in Figure 3. Note that in Figure 3, the `genView` and `plotView` classes are defined in the *MVCClass* package and the `graphView`, and `heatmapView` classes are defined in the *BioMVCClass* package.

The `genView` class has three slots, dataName, win, and winNum. dataName is the name of the model that the view displays, win is the Gtk window object that holds the view, and winNum is the number of the window so that the window can be identified in the GUI that other packages create. All views need to know these three pieces of information so `genView` binds the view classes together.

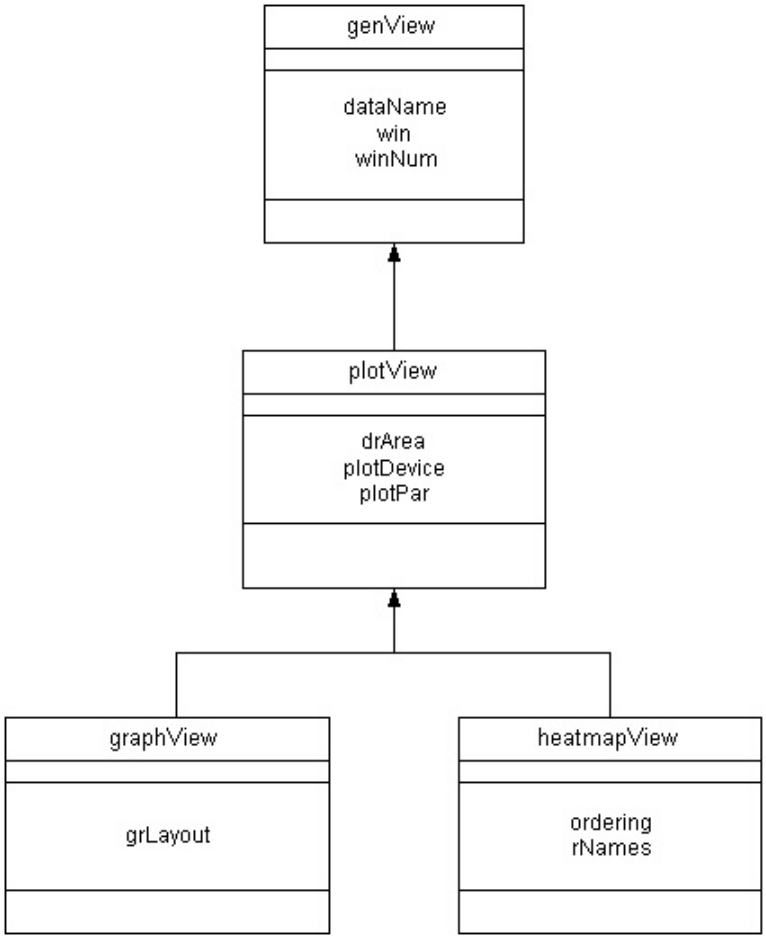

Figure 3: Inheritance for View Objects.

When creating a plot of a data set, there is a general class, called `plotView`, that has the slots, plotDevice, plotPar, and drArea, which store the device number of the plot, the plotting parameters, and the Gtk drawing area object, respectively. This class is a virtual class because it is not intended to have any objects as it just represents a general plot. The plot classes that can have objects are the specific plot classes, `graphView` and `heatmapView`, which inherit from the `plotView` class.

The `graphView` class represents a graph plot view and it has the extra slot, grLayout, which holds a `Ragraph` object that represents the layout of the graph plot. The `heatmapView` represents a heatmap view of the data and it contains the slots, ordering and rNames, which are a list that is returned from the `heatmap` function to give information about the dendrogram ordering and the names of the rows from the model that are shown in the heatmap, respectively. Note that some of these views are only applicable for certain model types. For example, the `graphView` class only makes sense for a view of a graph object, which would be stored in the `graphModel` class.

4 Conclusions

This package is intended to be a behind the scenes package that is used by other packages for its class definitions. Currently, this package is used by the *iSNetwork* package.