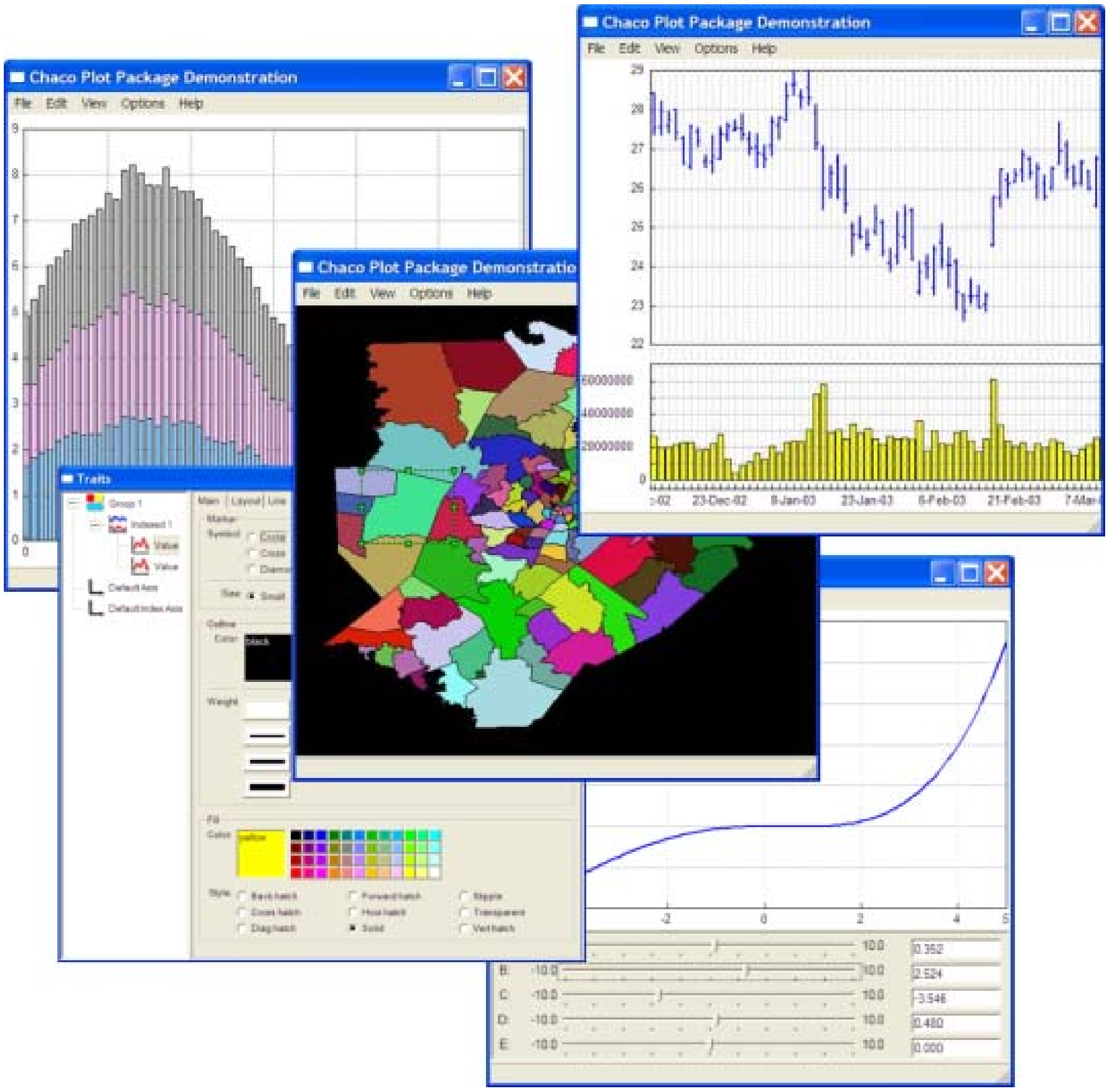
The Chaco User's Guide

David C. Morrill
Enthought, Inc.



Introduction

With packages such as Mathematica and MatLab, scientists and engineers already have a variety of high-quality plotting capabilities available to them. However, if you are a scientist or engineer who is also a Python user, or you are working with a limited budget, your choices are more limited. Chaco is an open source and freely available plotting package being developed to address many of the plotting needs of the scientific and engineering communities.

In many ways Python is a language well suited for scientific and engineering calculations, especially when used in conjunction with packages like Numeric or SciPy. Its high-level syntax and interactive shell, combined with the high-performance and easy to use array manipulation capabilities provided by Numeric or SciPy, make Python an attractive vehicle for scientific number crunching and analysis. Quite often the only thing missing from the "equation" is a high-quality, easy to use plotting package to express the results visually, either interactively on a computer screen, or as hard-copy via PDF or PostScript files.

Chaco is a package developed to fill that void, and to do so in a way that scales well from simple, ad-hoc plots done from the Python command shell to elaborate plotting applications. And of course, Chaco is fully buzzword compliant:

- Open source (covered by a BSD style license)
- Works with both the wxPython and Tkinter GUI toolkits
- Cross-platform (courtesy of Python, Numeric, SciPy, wxPython and Tkinter)
- Supports multiple output types (display, PDF, PostScript, GIF, PNG, JPEG)

Chaco can be used interactively, either through the Python command shell or as part of a larger wxPython or Tkinter based application. It can also be used non-interactively, for example within a CGI request handler from a web server to dynamically generate plot images for use on web pages.

The Core Plotting Objects

Plotting is accomplished in Chaco using a small set of core plotting objects:

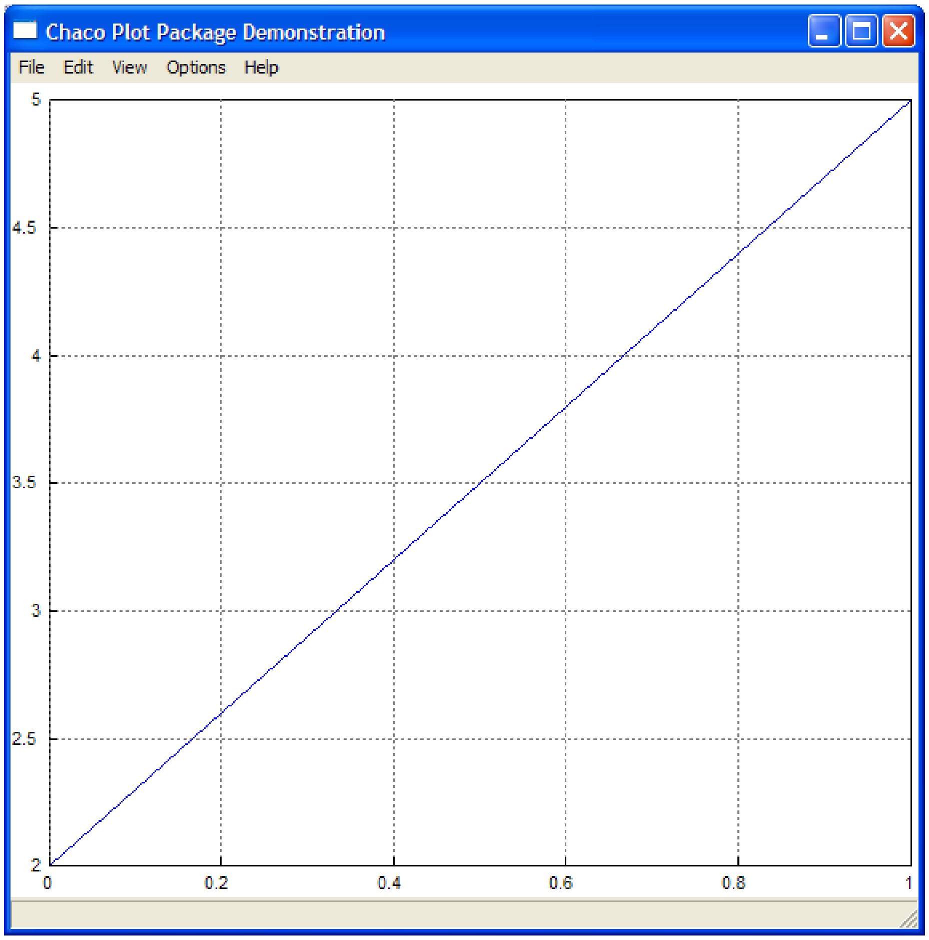
- **PlotValue**
- **PlotCanvas**
- **PlotIndexed**
- **PlotGroup**
- **PlotAxis**
- **PlotTitle**
- **PlotLabel**

For interactive use there is also an additional **PlotWindow** object that binds a collection of plotting objects to a particular GUI toolkit and display window.

Plots are created by composing plot objects in various ways. For example, perhaps the simplest possible plot is constructed using:

```
PlotValue( [ 2, 5 ] )
```

which creates a line plot of the two values (0,2) and (1,5) (the omitted x-values, 0 and 1, are automatically created by the **PlotValue** object). The following is an illustration of what this plot looks like using wxPython:



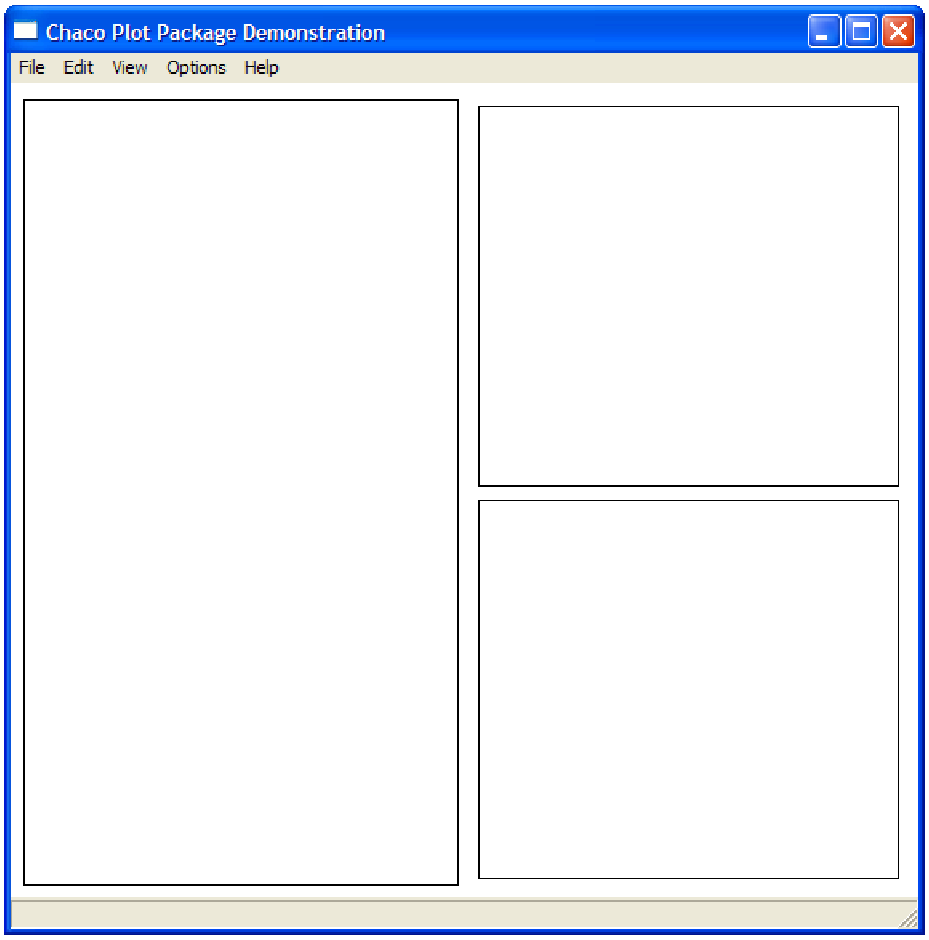
More complex plots are constructed by composing the core Chaco plotting objects in different ways.

In the following sections we will provide a high-level overview of the features and characteristics of each of the core classes, as well as information on how to construct them. In most cases, each section is also accompanied by a visual illustration of the object in actual use. Subsequent sections will describe the Chaco container and trait models and provide detailed information on the methods and traits available for each Chaco class.

The PlotGroup Class

A **PlotGroup** object is a container object used to contain and visually layout and organize other Chaco plotting objects. Any number of the other Chaco plotting objects can be added to a **PlotGroup**, including other **PlotGroup** objects. A **PlotGroup** has an **orientation** attribute, which can be either horizontal or vertical. Its orientation specifies the direction in which its contained objects are laid out.

This is illustrated in the following figure:



which shows a collection of three **PlotCanvas** objects organized using two nested **PlotGroup** objects:

```
PlotGroup(
   PlotCanvas(),
   PlotGroup(
      PlotCanvas(),
      PlotCanvas(),
      orientation = ‘vertical’ ),
   orientation = ‘horizontal’ )
```

The constructor for the **PlotGroup** class has the form:

PlotGroup([object, object, ...], [trait = value, trait = value, ...])
Creates an instance of the **PlotGroup** class. Each of the **object**s specified is added to the **PlotGroup** created. Any number of **PlotGroup** traits can also be set by specifying the appropriate trait and value (i.e. **trait = value**) pairs.

The PlotValue Class

A **PlotValue** object describes a single collection of related data to be plotted. In many ways, it can be considered to be simply an array of data values. The following is a simple example of how a **PlotValue** might be created:

```
from Numeric import *
x = arange( 0, 2 * pi, (2 * pi) / 100 )
value = PlotValue( sin( x ) )
```

In this case, the **PlotValue** object contains the value of ***sine*** for all values between 0 an 2**pi* over an evenly spaced group of 100 samples.

If used by itself, as in our earlier example, a **PlotValue** will render itself, along with its associated axes and grid. However, a **PlotValue** can also be imbedded in other objects, such as a **PlotCanvas**. In this case, a **PlotValue** object delegates many of its rendering functions up to its containing object. This will be illustrated in the next section.

The constructor for the **PlotValue** class has the form:

PlotValue (data = None, [trait = value, trait = value, ...])
Creates an instance of the **PlotValue** class. **Data** specifies the values to be associated with the **PlotValue** object. It can be a list, a Numeric array, a **PlotData** or **PlotSource** object, or **None**. Any number of **PlotValue** traits can also be set by specifying the appropriate trait and value (i.e. **trait = value**) pairs.

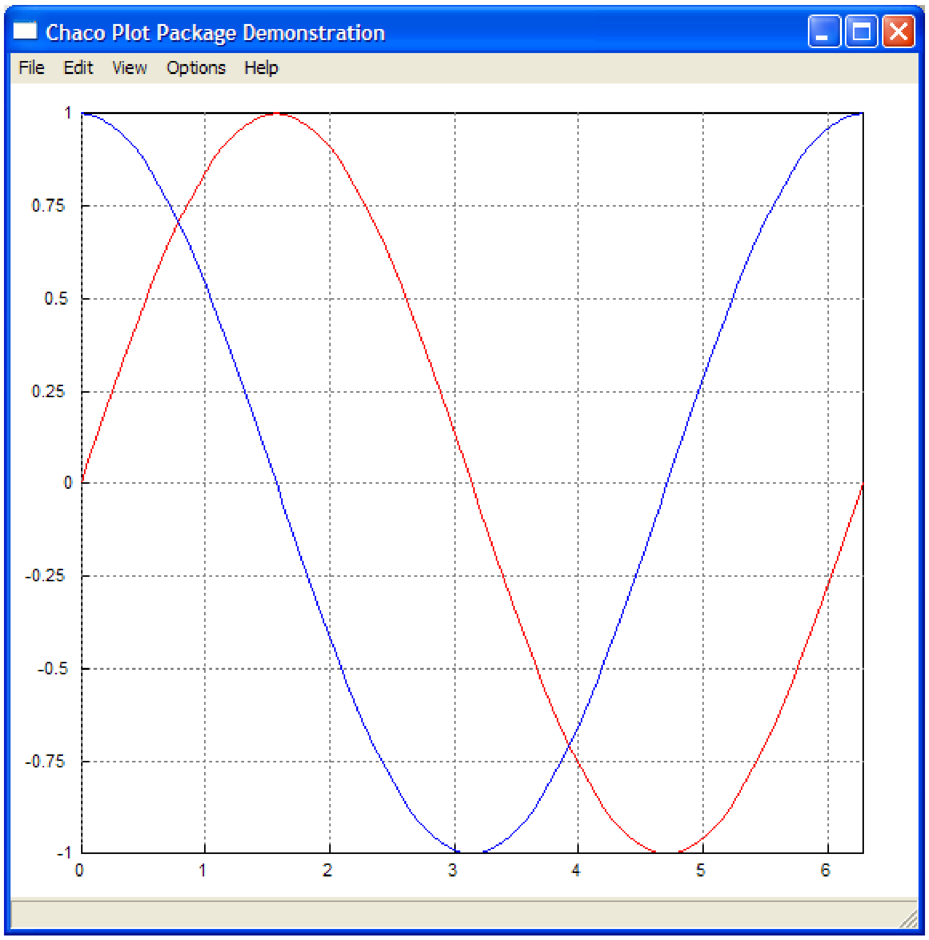
Each **PlotValue** *data* object also has associated with it a **PlotValue** *index* object. There are several ways of associating an index object with a data object:

- Do not specify any index information. In this case, an index equivalent to **PlotValue (arange(0.0, len(data)))** will be used.
- Set the data object's **index** trait using an appropriate **PlotValue** object.
- Specify a list or array of **(index, value)** pairs as the **data** argument to the **PlotValue** constructor. The constructor will automatically separate out the

index values and use them to create a **PlotValue** object assigned as the value of the constructed object's **index** trait.

The PlotCanvas Class

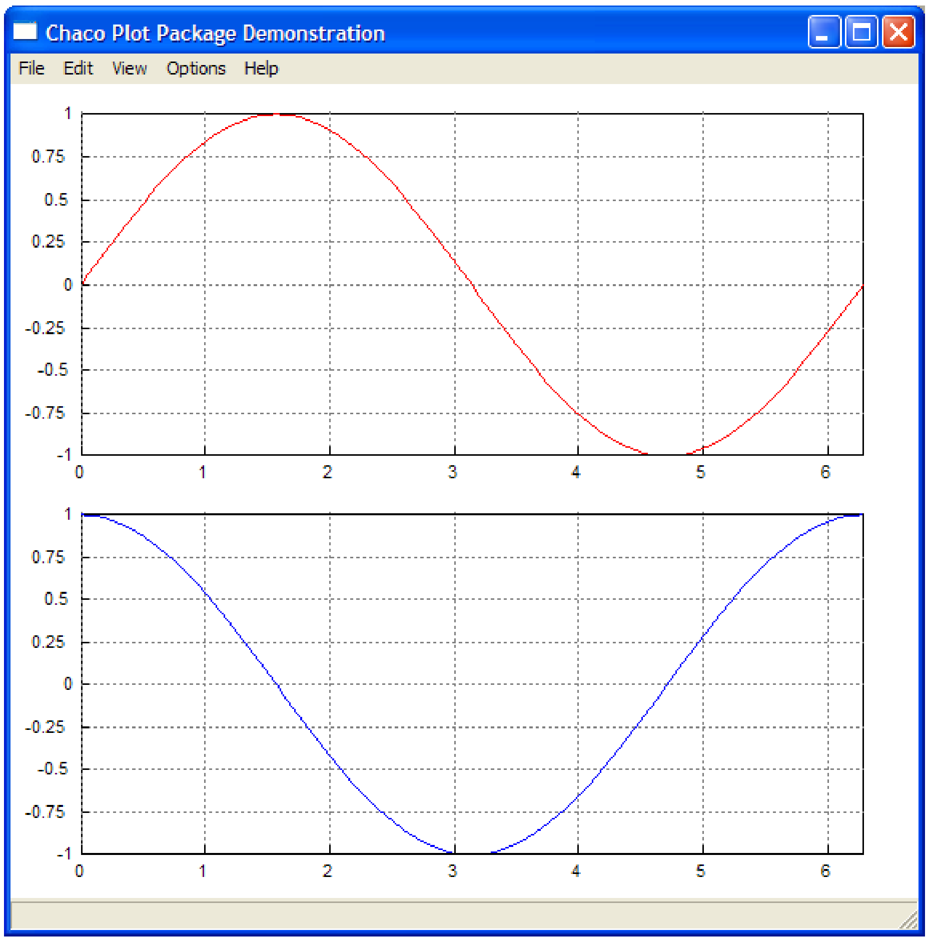
A **PlotCanvas** is, as its names suggests, a canvas upon which data is plotted. Like a **PlotGroup**, it can contain other Chaco plot objects. Unlike a **PlotGroup** however, a **PlotCanvas** does not lay out its component parts in separate, non-overlapping areas, but instead manages and organizes them so that they overlap in a meaningful and useful manner, as shown in the following figure:



which shows a **PlotCanvas** object containing two separate **PlotValue** objects, one containing the values of ***sine***, and the other containing the values of ***cosine***, over the interval from 0 to 2**pi*:

```
PlotCanvas(
   PlotValue( sin( arange( 0, 2 * pi, (2*pi) / 100 ) ),
              color = 'red' ),
   PlotValue( cos( arange( 0, 2 * pi, (2*pi) / 100 ) ),
              color = 'blue' )
)
```

Any number of **PlotValues** can be added to a single **PlotCanvas** object. Notice how the behavior of the **PlotCanvas** is different from that of a **PlotGroup**. If the same two **PlotValue** objects had been added to a **PlotGroup**, instead of a **PlotCanvas**, the result would have looked like:



The constructor for the **PlotCanvas** class has the form:

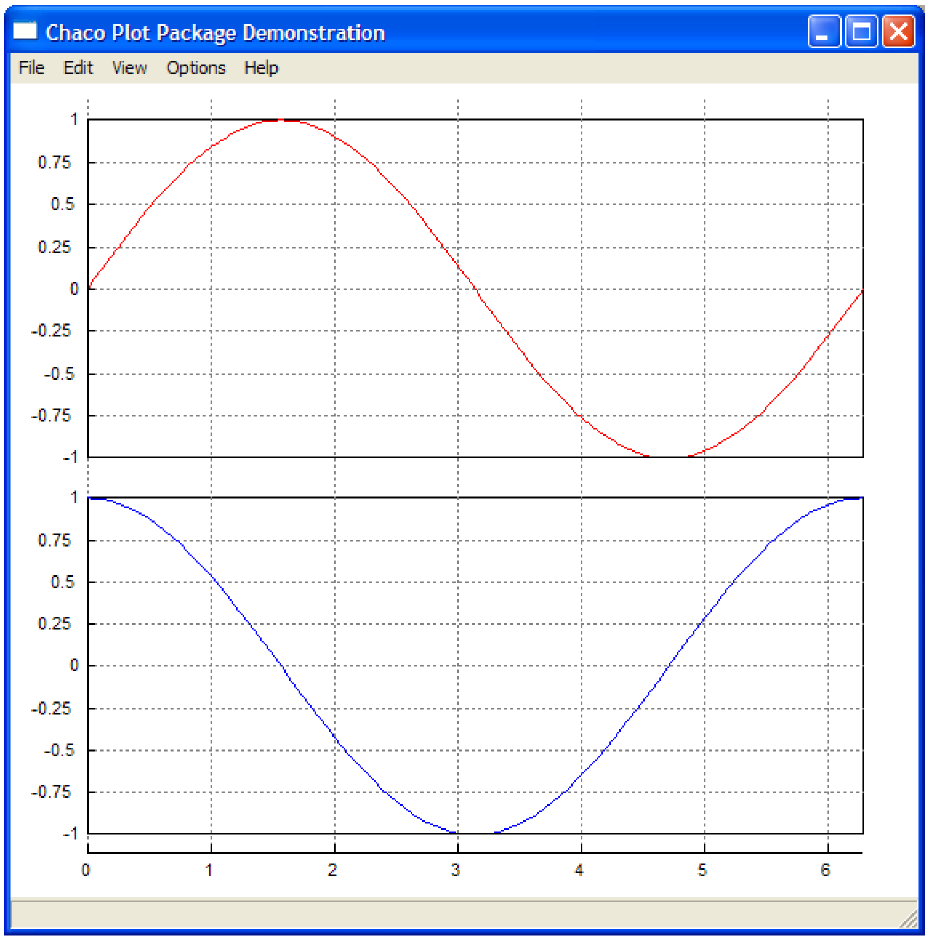
PlotCanvas([object, object, ...], [trait = value, trait = value, ...])
Creates an instance of the **PlotCanvas** class. Each of the **objects** specified is added to the **PlotCanvas** created. Any number of **PlotCanvas** traits can also be set by specifying the appropriate trait and value (i.e. **trait = value**) pairs.

The PlotIndexed Class

Another useful class that sits somewhere between the **PlotGroup** and **PlotCanvas** classes in terms of its layout and drawing capabilities is the **PlotIndexed** class. For the most part, a **PlotIndexed** object behaves very much like a **PlotGroup** object, laying out its contained objects in either a horizontal or vertical orientation. But unlike a **PlotGroup** object, which does not know anything about plotting, a **PlotIndexed** object understands plotting, and in fact factors out and coordinates the drawing of common index axes across all of its contained plottable objects.

What is an index axis? Because Chaco can plot data in any orientation, it would be awkward to talk in terms of the more standard mathematical *x* and *y* axes, since with a simple property change plots can easily be rotated 90 degrees or have the direction in which data is plotted be reversed. Instead, Chaco refers to a plot's *index* and *value* axes, where *value* describes the set of data being plotted, and corresponds in the normal case to the *y*-axis, and *index* describes the set of points at which the data was measured, and corresponds in the normal case to the *x*-axis.

So a **PlotIndexed** object factors out the common set of points at which various sets of data are measured and coordinates the drawing of all of the plottable objects so that they share a common index axis, as illustrated below:



Notice how in this case there is a single index (i.e. *x*) axis drawn along the bottom of the plot, and that the grid lines span the entire distance from top to bottom of the plotting area. **PlotIndexed** objects are very useful in cases where there are several sets of data derived from a common set of measurement points that need to be plotted together, but which may be too visually cluttered when drawn on a common **PlotCanvas** object.

The constructor for the **PlotIndexed** class has the form:

PlotIndexed([object, object, ...], [trait = value, trait = value, ...])
Creates an instance of the **PlotIndexed** class. Each of the **object**s specified is added to the **PlotIndexed** created. Any number of **PlotIndexed** traits can also be set by specifying the appropriate trait and value (i.e. **trait = value**) pairs.

The PlotAxis Class

A **PlotAxis** object describes the visual and layout characteristics of an axis. It controls such characteristics as the minimum and maximum plot bounds, the color and line styles used for the axis tick marks and grid lines, label formatting and spacing, and so on.

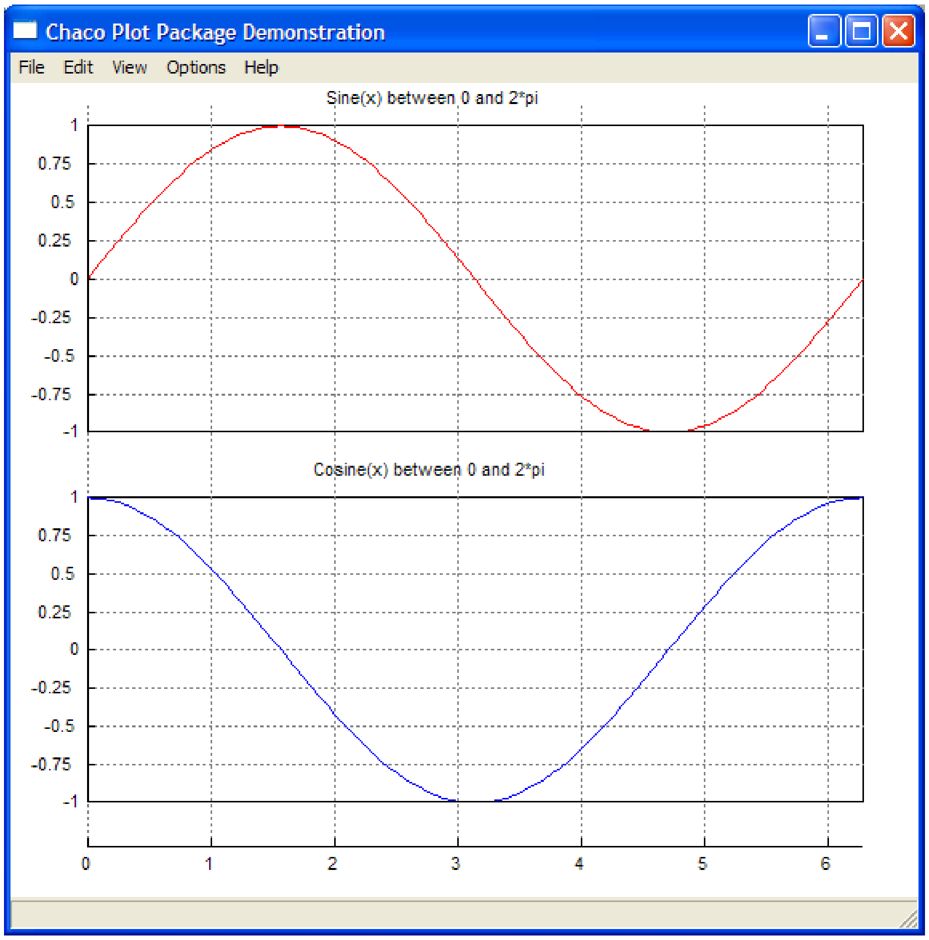
A **PlotAxis** is associated with a plottable object by setting either its **axis** or **axis_index** trait. The **axis_index** trait specifies the **PlotAxis** object to use when drawing the *index* axis of the object. The **axis** trait specifies the **PlotAxis** object to use when drawing the *value* axis of the object. If no **axis_index** trait has been specified for an object, the **PlotAxis** object associated with the **axis** trait will be used to draw the *index* axis.

The constructor for the **PlotAxis** class has the form:

PlotAxis([trait = value, trait = value, ...])
Creates an instance of the **PlotAxis** class. Any number of **PlotAxis** traits can also be set by specifying the appropriate trait and value (i.e. **trait = value**) pairs.

The PlotTitle Class

A **PlotTitle** object provides a way to add titles to any of the top-level Chaco plotting objects (i.e. **PlotGroup**, **PlotIndexed**, **PlotCanvas** and **PlotValue**). Any number of titles can be added to any of the four sides of any of these objects. The following shows the result of adding a **PlotTitle** object to the top of each of the **PlotValue** objects from the preceding figure:


Chaco Plot Package Demonstration
File Edit View Options Help
Sine(x) between 0 and 2*pi
1
0.75
0.5
0.25
0
-0.25
-0.5
-0.75
-1
Cosine(x) between 0 and 2*pi
1
0.75
0.5
0.25
0
-0.25
-0.5
-0.75
-1
0
1
2
3
4
5
6

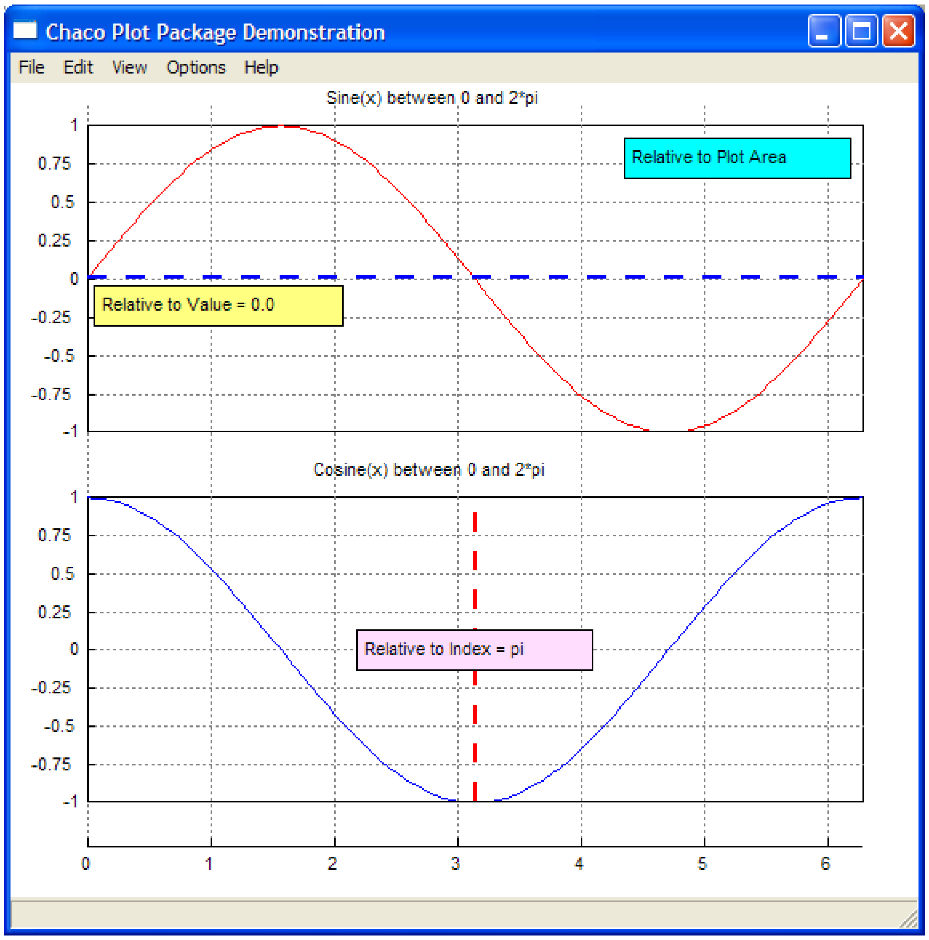
The constructor for the **PlotTitle** class has the form:

PlotTitle(text = None, [trait = value, trait = value, ...])

Creates an instance of the **PlotTitle** class. **Text** specifies the string to be used as the title's content (i.e. the value of the **PlotTitle**'s **text** trait). Any number of additional **PlotTitle** traits can also be set by specifying the appropriate trait and value (i.e. **trait = value**) pairs.

The PlotOverlay Class

A **PlotOverlay** object is similar to a **PlotTitle** object, but instead of being drawn outside the plot area, it overlays a section of the plotting area. A **PlotOverlay** object can either be positioned relative to the plotting area (e.g. 'top left') or relative to a particular **PlotValue** *index* or *data* value (or both). This is illustrated in the following figure, which shows examples of the different types of **PlotOverlay** objects:

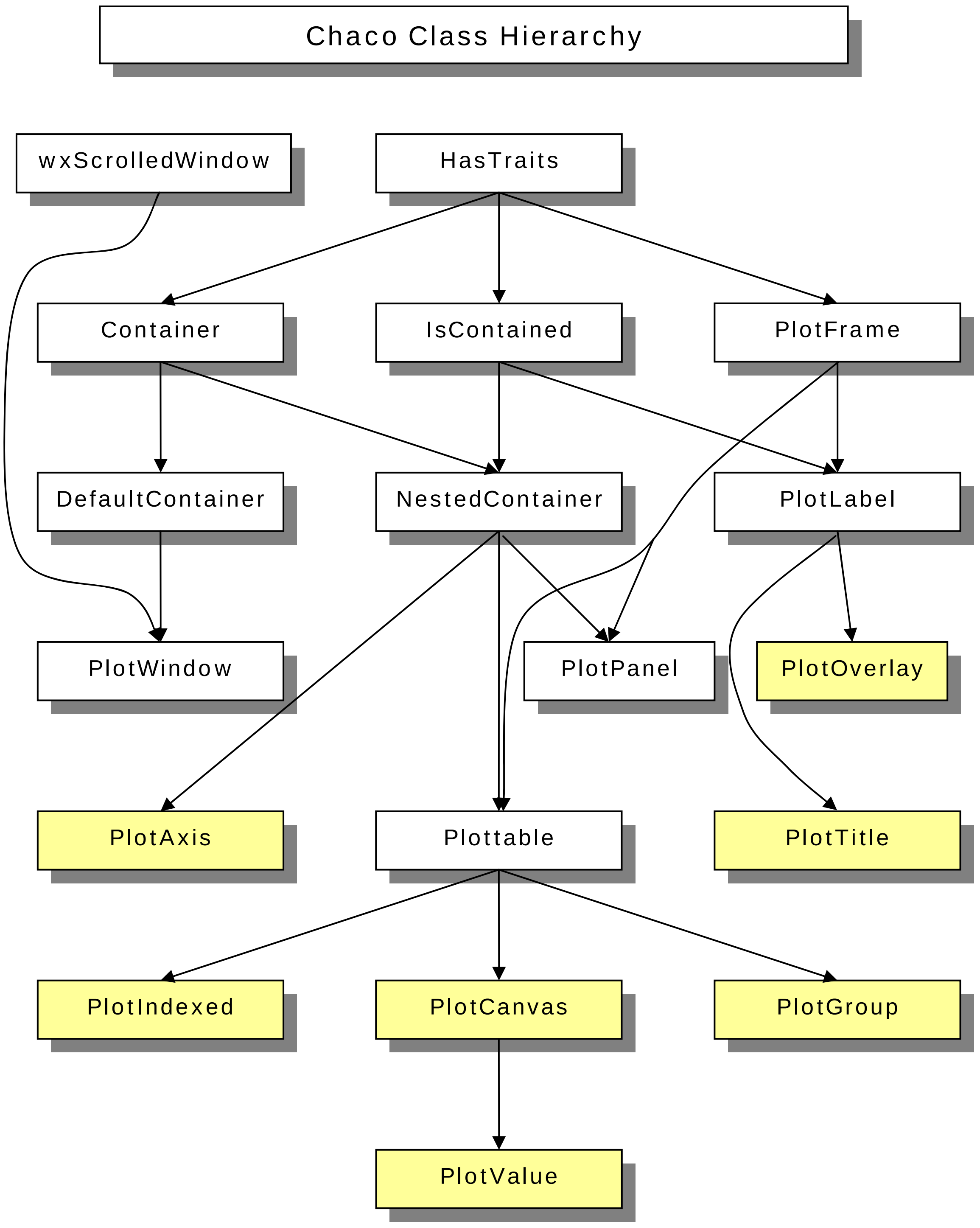

Chaco Plot Package Demonstration
File Edit View Options Help
Sine(x) between 0 and 2*pi
Relative to Plot Area
Relative to Value = 0.0
Cosine(x) between 0 and 2*pi
Relative to Index = pi
1
0.75
0.5
0.25
0
-0.25
-0.5
-0.75
-1
0
1
2
3
4
5
6

The constructor for the **PlotOverlay** class has the form:

PlotOverlay (text = None, [trait = value, trait = value, ...])
Creates an instance of the **PlotOverlay** class. **Text** specifies the string to be used as the overlay's content (i.e. the value of the **PlotOverlay**'s **text** trait). Any number of additional **PlotOverlay** traits can also be set by specifying the appropriate trait and value (i.e. **trait = value**) pairs.

The Chaco Class Hierarchy

So far we have provided a brief overview of most of the high-level Chaco plotting objects. The following figure illustrates these same classes in terms of their underlying Python class hierarchy:



Note that there are a number of classes that have not been mentioned so far. Some, like **Plottable** and **PlotFrame**, are base classes which define many of the common visual characteristics of the Chaco plotting objects we have seen in the preceding figures.

Others classes, like **IsContained**, **NestedContainer**, **Container** and **DefaultContainer** help define one of the other key architectural characteristics of Chaco: its container/delegation model.

The Chaco Container Model

All of the Chaco objects we have talked about so far can be characterized as objects that *can be contained* in other objects, or that *can contain* other objects, or both. The **IsContained** class is the base class for any Chaco object that can be contained in another object. The **NestedContainer** class is the base class for objects that can contain other objects. Since **NestedContainer** is a subclass of **IsContained**, objects derived from **NestedContainer** can both contain, and be contained in, other objects. Any object that can be contained in another object has an attribute called **container** that holds the reference to its container object.

The container model is used for two related, yet distinct, purposes in Chaco:

1. Containment is used to provide a visual hierarchy for Chaco objects. This is evident in the way that objects contained in **PlotGroup**, **PlotIndexed** and **PlotCanvas** objects are organized and drawn on the screen for example.

2. An object's container also provides a source of default values, such as line color, font, or axis orientation. If a particular object property is not explicitly set for an object, its value is automatically retrieved from its containing object. If the containing object does not have an explicit value set, the value is retrieved from its container, and so on, until a containing object which has an explicit value for the property is reached.

This second use of containment is referred to as delegation, and is based entirely on the **traits** package (refer to "The Traits User's Guide", available at **http://www.scipy.org/site_content/traits** for more information on the **traits** package).

Nearly every Chaco object property (i.e. *trait*) is defined using the delegation model. That is, every trait has a value defined either in the object itself, or in one of the container objects in its containment hierarchy. Note that the delegation chain is always guaranteed to terminate (that is, reach an object which has an explicit value set for the trait). This is true because of the **DefaultContainer** class, which defines explicit default values for every Chaco object trait.

The default value of every Chaco object's **container** trait is an instance of a **DefaultContainer** object. When a Chaco object is added to another Chaco container object, its **container** trait is modified to refer to its new container object. The topmost object in a Chaco object hierarchy is normally associated with a **PlotWindow** object. Referring to the previous class hierarchy figure, we see that **PlotWindow** is a subclass of **DefaultContainer**. Thus we are guaranteed that the delegation process always terminates in an instance of **DefaultContainer**, either the default one, or an instance of **PlotWindow**.

The delegation model is very useful because it allows global object defaults set on a top-level Chaco object to be automatically applied to all contained Chaco objects. Yet it still allows individual traits to be overridden as needed on any particular object. And because it is a hierarchy, defaults can be established at any level that apply to all subordinate objects.

The Container Classes

The four main Chaco container classes are:

- **PlotGroup**
- **PlotIndexed**
- **PlotCanvas**
- **PlotValue**

Instances of these classes can contain other Chaco objects. Objects are added to and removed from an object using the **add** and **remove** methods respectively. The **add** method has the form:

add(object, object, ...)
Adds one or more **objects** to the receiving container object. Each argument to **add** may be either a single object or list of objects to be added.

Each container object has several categories of objects that it understands. The **add** method examines each **object** being added and determines the category it belongs to. The possible categories are:

- Values
- Titles
- Overlays

If the **add** method of a container object does not recognize an object being added, it invokes the added object's **add_to_plot** method, passing it the container object as its argument. This allows new object types to be added to the Chaco toolkit without having to modify any of the container classes.

Note that when an object is added to a container object, it is automatically removed from the container, if any, it was previously contained in. Thus each object can be contained in at most one other object.

PlotTitle and **PlotOverlay** objects can be added to any of the Chaco container types, and will be added to their *titles* and *overlays* categories respectively.

A **PlotGroup** container can also have **PlotGroup**, **PlotIndexed**, **PlotCanvas** and **PlotValue** objects added to it. These will all be categorized as *values*, and will be the objects that the **PlotGroup** visually lays out.

A **PlotIndexed** container can have **PlotCanvas** and **PlotValue** objects added to it. These are categorized as *values*, and are the objects that the **PlotIndexed** container will lay out.

A **PlotCanvas** container can also have **PlotCanvas** and **PlotValue** objects added to it. Like a **PlotIndexed** object, these are also categorized as its *values*, and are the objects that the **PlotCanvas** container will lay out. Note that adding a **PlotCanvas** object to another **PlotCanvas** object is logically equivalent to adding the contents of the first **PlotCanvas** object to the second in terms of its visual effect.

A **PlotValue** object can only have *titles* and *overlays* added to it. It does not support adding any objects that could be categorized as *values*.

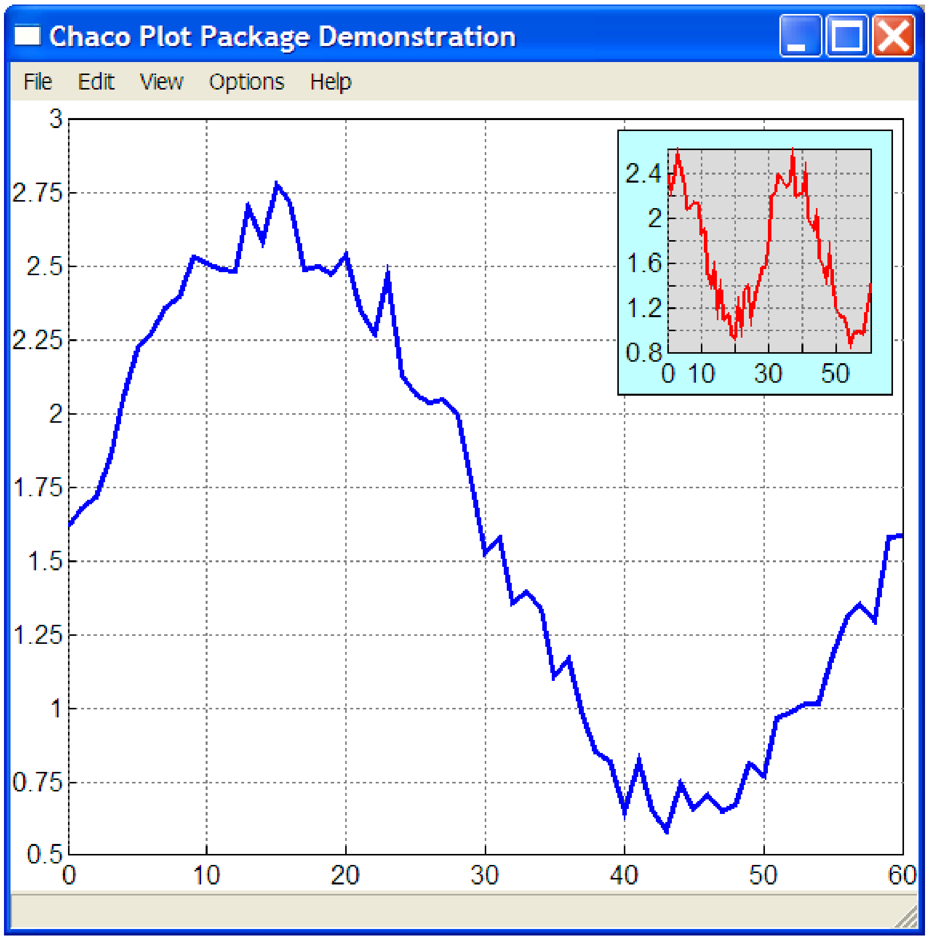
Each category of object that can be added to a Chaco container object can also be accessed directly using the container object's **values**, **titles**, or **overlays** traits. Each of these traits behaves exactly like a normal Python list, allowing indexing, iteration, slicing, insertion and so on. Thus the following two statements are identical in effect:

```
container.add( PlotOverlay(), PlotOverlay() )
container.overlays.extend([PlotOverlay(), PlotOverlay()])
```

Note that each list trait is strongly type checked, so attempting to add or insert an object of the wrong type will result in an exception.

Any Chaco container object can also be added to another Chaco container object as an *overlay* by setting its **overlay** trait to **true** before adding it. Just like a normal **PlotOverlay** object, the container object added as an overlay will be superimposed on top of the container object it is added to. This is illustrated in the following figure, which shows a **PlotCanvas** object that has been added to another **PlotCanvas** object using code similar to the following:

```
canvas.add( PlotCanvas( overlay = 'true' ) )
```



The **remove** method of a container object removes one or more objects from the container, and has the form:

remove(object, object, …)
Specifies one or more **objects** to be removed from the receiving container object. Each **object** specified in the argument list is removed from the container object. If an object is not in the container, it is ignored.

Objects can also be deleted from a container by performing list deletion or slice assignment on the appropriate container object's **values**, **titles**, or **overlays** traits. For example, the following code could be used to delete all objects currently contained in a container:

```
del container.values[:]
del container.titles[:]
del container.overlays[:]
```

There is also an alternate form of the **remove** method which has no arguments:

remove()

Removes the receiving object from its containing object. Note that this form of **remove** can also be used with non container objects such as **PlotTitle** and **PlotOverlay** objects.

Additional Container Methods

In general, most Chaco object functionality is accessed through traits, which will be discussed in the next section. However, there are a few additional methods available on container objects:

objects_at (x, y)

Returns the list of objects in the receiving container object that contain a point (**x**,**y**) specified in window coordinates. The list may be empty or contain one or more objects. More than one object is possible since objects can overlap.

The objects are ordered such that the first object in the list is *closest* to the user, and the last is *furthest* away.

Note that **PlotValue** objects contained within a **PlotCanvas** object are never returned by this method. Only the containing **PlotCanvas** object will be returned (if it contains the specified point). Use the **values_at** method to locate **PlotValues** that are near a specified point.

values_at (x, y)

Returns the list of **PlotValue** objects contained within the receiving object whose visual representation intersects (or is within a small delta of) a point (**x**,**y**) specified in window coordinates. The list may be empty or contain one or more objects. More than one object is possible since plots can overlap.

isin_plot (x, y)

Determines whether the point (**x**,**y**), specified in window coordinates, is contained within the plotting area of the receiving object, which must be either a **PlotCanvas** or **PlotValue** object. Returns 1 if the receiver's plotting area contains the specified point; and 0 otherwise.

The plotting area is the portion of a **PlotCanvas** (or **PlotValue**) where plotting occurs. It does not include the axes or titles area of the canvas.

The following methods only apply to **PlotValue** objects:

map_screen (index, value, index_offset = 0.0, value_offset = 0.0, xy = FALSE)

Maps **index** and **value** from data space to window space (i.e. window coordinates). **Index** specifies a point (or points) along the **PlotValue's** *index* axis, while **value** specifies a point (or points) along the **PlotValue's** *value* axis.

Index and **value** can be a scalar, an array, or **None**. If **index** or **value** is **None**, the corresponding part of the result is also **None**.

The **index_offset** and **value_offset** arguments allow an offset to be applied to the result *after* conversion to window coordinates. The default for both is 0.0.

The **xy** argument is a boolean value specifying whether or not the result is to be returned as separately mapped index and value coordinates (the default), or merged into combined (x,y) tuples (either a single tuple, or an array of tuples, depending upon whether **index** and **value** are scalars or arrays).

If **xy** is 0, the result returned is a tuple with the mapped index value(s) as the first element, and the mapped value value(s) as the second element.

If **xy** is non-zero, the result is an array of mapped (x,y) tuples. In this case, both **index** and **value** should both be scalars, or arrays with the same length.

map_xy (x, y)

Maps a point (**x**,**y**) specified in window coordinates to the corresponding point (x',y',v') in the data space of the **PlotValue** object (v' is the **PlotValue** intercept of a line orthogonal to the index axis that passes through the point (x',y')). The result returned is the mapped (x',y',v') tuple.

Note that in the case where the **PlotValue** object has multiple renderers, the first renderer specified is used to compute the mapping.

map_data (low, high = None)

Maps an index value (or range of index values) to the closest range of indices in the receiving **PlotValue** object's *index* data. **Low** specifies the lower bound of the range, and **high** specifies the upper bound. If **high** is omitted or **None**, it defaults to the same value as **low**.

The result is a tuple containing the integer indices of the elements in the **PlotValue's** index data that most closely match the range specified. If **low** or **high** lies between two values, the index of the element closest to the specified value is returned.

This method only returns meaningful results if the **PlotValue** object's index data is either monotonically increasing or decreasing. If this condition is not met, **(None, None)** is returned as the result.

The Trait Model

Nearly every characteristic of a Chaco object is defined as an object *trait* (using the **traits** package). This provides two major benefits:

1. It allows object properties to be defined in terms of the powerful containment and delegation model just described.

2. It provides a flexible, user friendly, yet type safe, method of setting object properties.

As an example of the flexibility provided by using traits to define object properties, consider the following example, which shows three different ways that a user can set the color of a **PlotTitle** object’s text:

```
title = PlotTitle( ‘My Plot’ )
title.color = 'blue'        # Standard color name
title.color = 0x0000FF      # Hexadecimal RGB calue
title.color = (0.,0.,1.,1.) # Floating point (R,G,B,Alpha)
```

Any of these are valid methods of setting the text color to blue. Note however that the following statement would generate an exception:

```
title.color = ‘pretty in pink’
```

because the string ‘pretty in pink’ is not recognized as one of the valid values for a Chaco color.

Setting Trait Values

There are two ways to set the value of a Chaco object trait:

1. Using standard Python object attribute assignment.
2. Using an object’s **set** method.

For example:

```
overlay.title = ‘Experimental Results’
overlay.color = ‘blue’
overlay.border_size = 1
```

is equivalent to:

```
overlay.set( title = ‘Experimental Results’,
             color = ‘blue’,
             border_size = 1 )
```

Chaco Object Trait Definitions

The following sections provide definitions and descriptions of the traits for each of the high-level Chaco plotting objects. Each section provides a brief description of the Chaco class followed by its trait definitions and descriptions organized into groups of related traits.

PlotTitle

A **PlotTitle** object is used to add a title to any of the four sides of a plot or container object. Any number of titles can be added to a **PlotGroup**, **PlotIndexed**, **PlotCanvas** or **PlotValue** object.

Title Related Traits

position	Specifies the side of the canvas the title appears on and should be **bottom** or **left** or **right** or **top** (or any unique prefix).

Label Related Traits

color	Specifies the title text color and should be a number, which in hex is of the form 0xrrggbb, where rr is red, gg is green, and bb is blue, or a sequence of 4 elements (rr, gg, bb, aa), where aa is alpha, with each element a number in the range from 0.0 to 1.0 or any standard color name.
text	Specifies the label text and should be of type **str**.
legend_color	Specifies the title legend color and should be a number, which in hex is of the form 0xrrggbb, where rr is red, gg is green, and bb is blue, or a sequence of 4 elements (rr, gg, bb, aa), where aa is alpha, with each element a number in the range from 0.0 to 1.0 or None or any standard color name.
font	Specifies the title text font and should be a Font or a string describing a font (e.g. **12 pt bold italic swiss family Arial** or **default 12**).
alignment	Specifies the alignment of the title text within its border and should be **bottom** or **center** or **left** or **right** or **top** (or any unique prefix).
rotate	Specifies the title rotation (in degrees) and should be 0 or 90.
visible	Specifies whether the title is visible and should be **f** or **false** or **n** or **no** or **off** or **on** or **t** or **true** or **y** or **yes** or 0 or 1.
debug	Specifies whether the margin and padding area bounding boxes should be drawn and should be **f** or **false** or **n** or **no** or **off** or **on** or **t** or **true** or **y** or **yes** or 0 or 1.

Frame Related Traits

bg_image	Specifies the background image and should be an Image or an image file name (e.g. a **.gif**, **.png** or **.jpeg** file) or **.**
bg_color	Specifies the title background color and should be a number, which in hex is of the form 0xrrggbb, where rr is red, gg is green, and bb is blue, or a sequence of 4 elements (rr, gg, bb, aa), where aa is alpha, with each element a number in the range from 0.0 to 1.0 or None or any standard color name.
border_color	Specifies the title border line color and should be a number, which in hex is

	of the form 0xrrggbb, where rr is red, gg is green, and bb is blue, or a sequence of 4 elements (rr, gg, bb, aa), where aa is alpha, with each element a number in the range from 0.0 to 1.0 or any standard color name.
border_size	Specifies the title border line thickness (in pixels) and should be an integer in the range from 0 to 15.
border_style	Specifies the title border line style and should be **bevel** or **drop shadow** or **solid** (or any unique prefix).
margin	Specifies the margin space outside the title border (in pixels) and should be an integer in the range from 0 to 31.
margin_height	Specifies the margin space outside the top and bottom sides of the title border (in pixels) and should be an integer in the range from 0 to 31.
margin_width	Specifies the margin space outside the left and right sides of the title border (in pixels) and should be an integer in the range from 0 to 31.
padding	Specifies the padding space inside the title border (in pixels) and should be an integer in the range from 0 to 31.
padding_height	Specifies the padding space inside the top and bottom sides of the title border (in pixels) and should be an integer in the range from 0 to 31.
padding_width	Specifies the padding space inside the left and right sides of the title border (in pixels) and should be an integer in the range from 0 to 31.

General Chaco Object Related Traits

label	Specifies a brief description of the object and should be of type **str**.
selected	Specifies whether the select markers should be drawn and should be **f** or **false** or **n** or **no** or **off** or **on** or **t** or **true** or **y** or **yes** or 0 or 1.
container	Specifies the object containing this object and should be a Container.
menu	Specifies the title pop-up menu and should be of type **str**.

PlotOverlay

A **PlotOverlay** object is used to add an image or text that overlays part of another Chaco object. Overlays can either be positioned relative to the object they are contained in or, in the case of a **PlotOverlay** object contained in a **Plotvalue** object, relative to a data or index value (or both). Any number of overlays can be added to a **PlotGroup**, **PlotIndexed**, **PlotCanvas** or **PlotValue** object.

Overlay Related Traits

position	Specifies the overlay's position relative to its container and should be **b** or **bl** or **bottom center** or **bottom left** or **bottom right** or **bottom** or **br** or **c** or **center bottom** or **center left** or **center right** or **center top** or **center** or **l** or **lb** or **left bottom** or **left center** or **left top** or **left** or **lt** or **r** or **rb** or **right bottom** or **right center** or **right top** or **right** or **rt** or **t** or **tl** or **top center** or **top left** or **top right** or **top** or **tr**.
value	Specifies the overlay value coordinate and should be of type **float** or None.
index	Specifies the overlay index coordinate and should be of type **float** or None.
line_style	Specifies the overlay line style and should be **dash** or **dot dash** or **dot** or **long dash** or **solid** (or any unique prefix).
line_color	Specifies the overlay line color and should be a number, which in hex is of the form 0xrrggbb, where rr is red, gg is green, and bb is blue, or a sequence of 4 elements (rr, gg, bb, aa), where aa is alpha, with each element a number in the range from 0.0 to 1.0 or None or any standard color name.
line_weight	Specifies the overlay line weight (in pixels) and should be an integer in the range from 1 to 9.

Label Related Traits

color	Specifies the overlay text color and should be a number, which in hex is of the form 0xrrggbb, where rr is red, gg is green, and bb is blue, or a sequence of 4 elements (rr, gg, bb, aa), where aa is alpha, with each element a number in the range from 0.0 to 1.0 or any standard color name.
text	Specifies the label text and should be of type **str**.
legend_color	Specifies the overlay legend color and should be a number, which in hex is of the form 0xrrggbb, where rr is red, gg is green, and bb is blue, or a sequence of 4 elements (rr, gg, bb, aa), where aa is alpha, with each element a number in the range from 0.0 to 1.0 or None or any standard color name.
font	Specifies the overlay text font and should be a Font or a string describing a font (e.g. **12 pt bold italic swiss family Arial** or **default 12**).
alignment	Specifies the alignment of the overlay text within its border and should be **bottom** or **center** or **left** or **right** or **top** (or any unique prefix).
rotate	Specifies the overlay rotation (in degrees) and should be 0 or 90.

visible	Specifies whether the overlay is visible and should be **f** or **false** or **n** or **no** or **off** or **on** or **t** or **true** or **y** or **yes** or 0 or 1.
debug	Specifies whether the margin and padding area bounding boxes should be drawn and should be **f** or **false** or **n** or **no** or **off** or **on** or **t** or **true** or **y** or **yes** or 0 or 1.

Frame Related Traits

bg_image	Specifies the background image and should be an Image or an image file name (e.g. a **.gif**, **.png** or **.jpeg** file) or **.**
bg_color	Specifies the overlay background color and should be a number, which in hex is of the form 0xrrggbb, where rr is red, gg is green, and bb is blue, or a sequence of 4 elements (rr, gg, bb, aa), where aa is alpha, with each element a number in the range from 0.0 to 1.0 or None or any standard color name.
border_color	Specifies the overlay border line color and should be a number, which in hex is of the form 0xrrggbb, where rr is red, gg is green, and bb is blue, or a sequence of 4 elements (rr, gg, bb, aa), where aa is alpha, with each element a number in the range from 0.0 to 1.0 or any standard color name.
border_size	Specifies the overlay border line thickness (in pixels) and should be an integer in the range from 0 to 15.
border_style	Specifies the overlay border line style and should be **bevel** or **drop shadow** or **solid** (or any unique prefix).
margin	Specifies the margin space outside the overlay border (in pixels) and should be an integer in the range from 0 to 31.
margin_height	Specifies the margin space outside the top and bottom sides of the overlay border (in pixels) and should be an integer in the range from 0 to 31.
margin_width	Specifies the margin space outside the left and right sides of the overlay border (in pixels) and should be an integer in the range from 0 to 31.
padding	Specifies the padding space inside the overlay border (in pixels) and should be an integer in the range from 0 to 31.
padding_height	Specifies the padding space inside the top and bottom sides of the overlay border (in pixels) and should be an integer in the range from 0 to 31.
padding_width	Specifies the padding space inside the left and right sides of the overlay border (in pixels) and should be an integer in the range from 0 to 31.

General Chaco Object Related Traits

label	Specifies a brief description of the object and should be of type **str**.
selected	Specifies whether the select markers should be drawn and should be **f** or **false** or **n** or **no** or **off** or **on** or **t** or **true** or **y** or **yes** or 0 or 1.
container	Specifies the object containing this object and should be a Container.

menu	Specifies the overlay pop-up menu and should be of type **str**.

PlotAxis

A **PlotAxis** object describes the visual and layout characteristics of a plottable object's axes. A single **PlotAxis** object can be used with any number of plottable objects to describe their index and/or value axes.

A **PlotAxis** object is associated with the **value** axis of an object by assigning it as the value of the plottable object's **axis** trait. It can be associated with an object's **index** axis by assigning it as the value of the object's **axis_index** trait.

If no **PlotAxis** object has been assigned to a an object's **axis_index** trait, the **PlotAxis** object assigned to the object's **axis** trait is used.

Axis Related Traits

title	Specifies None and should be a PlotLabel or of type **str**.
scale	Specifies the scaling mode of the axis and should be **linear** or **log** (or any unique prefix).
position	Specifies the sides of the plot the axis appears on and should be **all** or **both** or **bottom** or **left** or **right** or **top** (or any unique prefix).
line_weight	Specifies the axis line weight (in pixels) and should be an integer in the range from 1 to 9.
line_color	Specifies the axis line color and should be a number, which in hex is of the form 0xrrggbb, where rr is red, gg is green, and bb is blue, or a sequence of 4 elements (rr, gg, bb, aa), where aa is alpha, with each element a number in the range from 0.0 to 1.0 or any standard color name.
line_style	Specifies the axis line style and should be **dash** or **dot dash** or **dot** or **long dash** or **solid** (or any unique prefix).
label_font	Specifies the axis label font and should be a Font or a string describing a font (e.g. **12 pt bold italic swiss family Arial** or **default 12**).
label_color	Specifies the axis label text color and should be a number, which in hex is of the form 0xrrggbb, where rr is red, gg is green, and bb is blue, or a sequence of 4 elements (rr, gg, bb, aa), where aa is alpha, with each element a number in the range from 0.0 to 1.0 or any standard color name.
label_style	Specifies the location of labels along the axis and should be an integer in the range from 1 to 10 or **all** or **edge** or **end** or **none** (or any unique prefix).
label_offset	Specifies the offset of axis labels from the tick marks and should be an integer in the range from 0 to 31.
label_format	Specifies the format to apply when creating axis labels and should be a standard Python conversion string (e.g. **%%.2f**) or a function which converts a number to a string.
tick_visible	Specifies whether the axis tick marks are visible and should be **f** or **false** or **n** or **no** or **off** or **on** or **t** or **true** or **y** or **yes** or 0 or 1.

tick_weight	Specifies the axis tick mark line weight (in pixels) and should be an integer in the range from 1 to 9.
tick_color	Specifies the axis tick mark color and should be a number, which in hex is of the form 0xrrggbb, where rr is red, gg is green, and bb is blue, or a sequence of 4 elements (rr, gg, bb, aa), where aa is alpha, with each element a number in the range from 0.0 to 1.0 or any standard color name.
tick_style	Specifies the axis tick mark line style and should be **dash** or **dot dash** or **dot** or **long dash** or **solid** (or any unique prefix).
tick_in	Specifies the length of axis ticks into the plot (in pixels) and should be an integer in the range from 0 to 31.
tick_out	Specifies the length of axis ticks out from the plot (in pixels) and should be an integer in the range from 0 to 31.
tick_interval	Specifies the spacing of axis tick marks and should be **auto** (or any unique prefix) or a floating point number in the range from 0.0 to 1e+308 or an integer in the range from -100 to -1.
grid_visible	Specifies whether the axis grid lines are visible and should be **f** or **false** or **n** or **no** or **off** or **on** or **t** or **true** or **y** or **yes** or 0 or 1.
grid_weight	Specifies the axis grid line weight (in pixels) and should be an integer in the range from 1 to 9.
grid_color	Specifies the axis grid line color and should be a number, which in hex is of the form 0xrrggbb, where rr is red, gg is green, and bb is blue, or a sequence of 4 elements (rr, gg, bb, aa), where aa is alpha, with each element a number in the range from 0.0 to 1.0 or any standard color name.
grid_style	Specifies the axis grid line style and should be **dash** or **dot dash** or **dot** or **long dash** or **solid** (or any unique prefix).
margin	Specifies the margin space around the outside of the axis (in pixels) and should be an integer in the range from 0 to 31.
margin_height	Specifies the margin space outside the top and bottom sides of the axis (in pixels) and should be an integer in the range from 0 to 31.
margin_width	Specifies the margin space outside the left and right sides of the axis (in pixels) and should be an integer in the range from 0 to 31.
bound_low	Specifies the lower axis bound and should be **auto** or **fit** (or any unique prefix) or of type **float**.
bound_high	Specifies the upper axis bound and should be **auto** or **fit** (or any unique prefix) or of type **float**.
zoomable	Specifies whether the axis can be drag zoomed by the user and should be **f** or **false** or **n** or **no** or **off** or **on** or **t** or **true** or **y** or **yes** or 0 or 1.

General Chaco Object Related Traits

label	Specifies a brief description of the object and should be of type **str**.

selected	Specifies whether the select markers should be drawn and should be **f** or **false** or **n** or **no** or **off** or **on** or **t** or **true** or **y** or **yes** or 0 or 1.
container	Specifies the object containing this object and should be a Container.
debug	Specifies whether the margin and padding area bounding boxes should be drawn and should be **f** or **false** or **n** or **no** or **off** or **on** or **t** or **true** or **y** or **yes** or 0 or 1.
visible	Specifies whether the axis is visible and should be **f** or **false** or **n** or **no** or **off** or **on** or **t** or **true** or **y** or **yes** or 0 or 1.
menu	Specifies the axis pop-up menu and should be of type **str**.

PlotCanvas

A **PlotCanvas** object plots one or more **PlotValue** objects on a single plotting area, called a *canvas*. Besides **PlotValue** objects, a **PlotCanvas** object can also contain **PlotTitle** and **PlotOverlay** objects, as well as **PlotGroup**, **PlotIndexed**, **PlotCanvas** and **PlotValue** objects whose **overlay** trait value is **true**.

Note that a **PlotCanvas** object can also contain other **PlotCanvas** objects whose **overlay** trait value is not **true**. In this case, each **PlotValue** or **PlotCanvas** value contained in the nested **PlotCanvas** object is handled visually as if it were contained in the outermost **PlotCanvas** object.

A **PlotCanvas** object can be added to a **PlotGroup** or **PlotIndexed** object.

Canvas Related Traits

index_scrollbar	Specifies the scrollbar to use for scrolling the index and should be None.
value_scrollbar	Specifies the scrollbar to use for scrolling the value and should be None.

Plottable Object Related Traits

titles	Specifies None and should be any value.
overlays	Specifies None and should be any value.
values	Specifies None and should be any value.
overlay	Specifies whether the canvas overlays its container and should be **f** or **false** or **n** or **no** or **off** or **on** or **t** or **true** or **y** or **yes** or 0 or 1.
position	Specifies the canvas's position relative to its container and should be **b** or **bl** or **bottom center** or **bottom left** or **bottom right** or **bottom** or **br** or **c** or **center bottom** or **center left** or **center right** or **center top** or **center** or **l** or **lb** or **left bottom** or **left center** or **left top** or **left** or **lt** or **r** or **rb** or **right bottom** or **right center** or **right top** or **right** or **rt** or **t** or **tl** or **top center** or **top left** or **top right** or **top** or **tr**.
index_scale	Specifies the index scaling factor and should be a floating point number in the range from 1.0 to 100.0.
value_scale	Specifies the value scaling factor and should be a floating point number in the range from 1.0 to 100.0.
cursor_type	Specifies the plotting area cursor drawing type and should be **both** or **index** or **none** or **value** (or any unique prefix).
cursor_color	Specifies the cursor line color and should be a number, which in hex is of the form 0xrrggbb, where rr is red, gg is green, and bb is blue, or a sequence of 4 elements (rr, gg, bb, aa), where aa is alpha, with each element a number in the range from 0.0 to 1.0 or any standard color name.
cursor_style	Specifies the cursor line style and should be **dash** or **dot dash** or **dot** or **long**

	dash or **solid** (or any unique prefix).
cursor_weight	Specifies the cursor line weight (in pixels) and should be an integer in the range from 1 to 9.
index	Specifies None and should be a PlotValue or None.
index_axis	Specifies the axis the index axis is aligned with and should be **x** or **y**.
axis	Specifies None and should be a PlotAxis.
axis_index	Specifies None and should be a PlotAxis or None.

Frame Related Traits

bg_image	Specifies the background image and should be an Image or an image file name (e.g. a **.gif**, **.png** or **.jpeg** file) or **.**
bg_color	Specifies the canvas background color and should be a number, which in hex is of the form 0xrrggbb, where rr is red, gg is green, and bb is blue, or a sequence of 4 elements (rr, gg, bb, aa), where aa is alpha, with each element a number in the range from 0.0 to 1.0 or None or any standard color name.
border_color	Specifies the canvas border line color and should be a number, which in hex is of the form 0xrrggbb, where rr is red, gg is green, and bb is blue, or a sequence of 4 elements (rr, gg, bb, aa), where aa is alpha, with each element a number in the range from 0.0 to 1.0 or any standard color name.
border_size	Specifies the canvas border line thickness (in pixels) and should be an integer in the range from 0 to 15.
border_style	Specifies the canvas border line style and should be **bevel** or **drop shadow** or **solid** (or any unique prefix).
margin	Specifies the margin space outside the canvas border (in pixels) and should be an integer in the range from 0 to 31.
margin_height	Specifies the margin space outside the top and bottom sides of the canvas border (in pixels) and should be an integer in the range from 0 to 31.
margin_width	Specifies the margin space outside the left and right sides of the canvas border (in pixels) and should be an integer in the range from 0 to 31.
padding	Specifies the padding space inside the canvas border (in pixels) and should be an integer in the range from 0 to 31.
padding_height	Specifies the padding space inside the top and bottom sides of the canvas border (in pixels) and should be an integer in the range from 0 to 31.
padding_width	Specifies the padding space inside the left and right sides of the canvas border (in pixels) and should be an integer in the range from 0 to 31.

Container Related Traits

min_height	Specifies the minimum height of the canvas (in pixels) and should be an integer in the range from 0 to 99999.
min_width	Specifies the minimum width of the canvas (in pixels) and should be an integer in the range from 0 to 99999.
weight	Specifies the weight of the canvas's size relative to other canvas's in the same container and should be a floating point number in the range from 0.0 to 10.0.

General Chaco Object Related Traits

container	Specifies the object containing this object and should be a Container.
selected	Specifies whether the select markers should be drawn and should be **f** or **false** or **n** or **no** or **off** or **on** or **t** or **true** or **y** or **yes** or 0 or 1.
label	Specifies a brief description of the object and should be of type **str**.
debug	Specifies whether the margin and padding area bounding boxes should be drawn and should be **f** or **false** or **n** or **no** or **off** or **on** or **t** or **true** or **y** or **yes** or 0 or 1.
visible	Specifies whether the canvas object is visible and should be **f** or **false** or **n** or **no** or **off** or **on** or **t** or **true** or **y** or **yes** or 0 or 1.
menu	Specifies the canvas pop-up menu and should be of type **str**.

PlotValue

A **PlotValue** object represents an array of related, plottable data. It can either be plotted by itself, or added to a **PlotCanvas** object where it can be plotted along with other **PlotValue** objects on a single plotting area, or *canvas*.

PlotValue objects can also be added to **PlotGroup** and **PlotIndexed** objects, although in these cases the plot will not overlap any other plotting area.

A **PlotValue** can also be added to a **PlotValue**, **PlotCanvas**, **PlotGroup** or **PlotIndexed** object with its **overlay** trait set to **true**. In this case, the **PlotValue** will be drawn as a separated plot overlaying the object it is added to.

Value Related Traits

data	Specifies the data to be plotted and should be a sequence or tuple of numbers, a PlotSource object, or a Numeric array whose shape is either (n), (n,2) or (2,n).
origin	Specifies the corner of the plotting area used as the axis origin and should be **bl** or **bottom left** or **bottom right** or **br** or **lb** or **left bottom** or **left top** or **lt** or **rb** or **right bottom** or **right top** or **rt** or **tl** or **top left** or **top right** or **tr**.
type	Specifies the type of plot drawn and should be a comma separated plot type string (e.g. **line,scatter**) or plot type list (e.g. ['line,'scatter']).
line_style	Specifies the plot line style and should be **dash** or **dot dash** or **dot** or **long dash** or **solid** (or any unique prefix).
line_color	Specifies the plot line color and should be a number, which in hex is of the form 0xrrggbb, where rr is red, gg is green, and bb is blue, or a sequence of 4 elements (rr, gg, bb, aa), where aa is alpha, with each element a number in the range from 0.0 to 1.0 or None or any standard color name.
line_weight	Specifies the plot line weight (in pixels) and should be an integer in the range from 1 to 9.
outline_color	Specifies the plot marker outline color and should be a number, which in hex is of the form 0xrrggbb, where rr is red, gg is green, and bb is blue, or a sequence of 4 elements (rr, gg, bb, aa), where aa is alpha, with each element a number in the range from 0.0 to 1.0 or any standard color name.
outline_weight	Specifies the plot marker outline weight (in pixels) and should be an integer in the range from 0 to 9.
fill_color	Specifies the plot marker fill color and should be a number, which in hex is of the form 0xrrggbb, where rr is red, gg is green, and bb is blue, or a sequence of 4 elements (rr, gg, bb, aa), where aa is alpha, with each element a number in the range from 0.0 to 1.0 or any standard color name.
fill_style	Specifies the plot marker fill style and should be **back hatch** or **cross hatch** or **diag hatch** or **forward hatch** or **horz hatch** or **solid** or **stipple** or **transparent** or **vert hatch** (or any unique prefix).

size	Specifies the plot marker size and should be **large** or **medium** or **small** (or any unique prefix).
symbol	Specifies the plot marker symbol type and should be **circle** or **square** or **diamond** or **triangle** or **down triangle** or **cross** or **plus** (or any unique prefix).
bar_width	Specifies the width of a plot bar as a percentage of the available space and should be a floating point number in the range from 10.0 to 150.0.
bin_width	Specifies the width of a group of plot bars as a percentage of the available space and should be a floating point number in the range from 50.0 to 100.0.
select_type	Specifies the type of range the user can drag select and should be **index** or **none** or **value** (or any unique prefix).
select_notify	Specifies the routine to be called when a drag select operation is complete and should be of type **function** or of type **instance method** or None.
index_scrollbar	Specifies the scrollbar to use for scrolling the index and should be None.
value_scrollbar	Specifies the scrollbar to use for scrolling the value and should be None.

Plottable Object Related Traits

titles	Specifies None and should be any value.
overlays	Specifies None and should be any value.
values	Specifies None and should be any value.
overlay	Specifies whether the canvas overlays its container and should be **f** or **false** or **n** or **no** or **off** or **on** or **t** or **true** or **y** or **yes** or 0 or 1.
position	Specifies the canvas's position relative to its container and should be **b** or **bl** or **bottom center** or **bottom left** or **bottom right** or **bottom** or **br** or **c** or **center bottom** or **center left** or **center right** or **center top** or **center** or **l** or **lb** or **left bottom** or **left center** or **left top** or **left** or **lt** or **r** or **rb** or **right bottom** or **right center** or **right top** or **right** or **rt** or **t** or **tl** or **top center** or **top left** or **top right** or **top** or **tr**.
index_scale	Specifies the index scaling factor and should be a floating point number in the range from 1.0 to 100.0.
value_scale	Specifies the value scaling factor and should be a floating point number in the range from 1.0 to 100.0.
cursor_type	Specifies the plotting area cursor drawing type and should be **both** or **index** or **none** or **value** (or any unique prefix).
cursor_color	Specifies the cursor line color and should be a number, which in hex is of the form 0xrrggbb, where rr is red, gg is green, and bb is blue, or a sequence of 4 elements (rr, gg, bb, aa), where aa is alpha, with each element a number in the range from 0.0 to 1.0 or any standard color name .
cursor_style	Specifies the cursor line style and should be **dash** or **dot dash** or **dot** or **long dash** or **solid** (or any unique prefix).
cursor_weight	Specifies the cursor line weight (in pixels) and should be an integer in the

	range from 1 to 9.
index	Specifies None and should be a PlotValue or None.
index_axis	Specifies the axis the index axis is aligned with and should be **x** or **y**.
axis	Specifies None and should be a PlotAxis.
axis_index	Specifies None and should be a PlotAxis or None.

Frame Related Traits

bg_image	Specifies the background image and should be an Image or an image file name (e.g. a **.gif**, **.png** or **.jpeg** file) or **.**
bg_color	Specifies the canvas background color and should be a number, which in hex is of the form 0xrrggbb, where rr is red, gg is green, and bb is blue, or a sequence of 4 elements (rr, gg, bb, aa), where aa is alpha, with each element a number in the range from 0.0 to 1.0 or None or any standard color name .
border_color	Specifies the canvas border line color and should be a number, which in hex is of the form 0xrrggbb, where rr is red, gg is green, and bb is blue, or a sequence of 4 elements (rr, gg, bb, aa), where aa is alpha, with each element a number in the range from 0.0 to 1.0 or any standard color name.
border_size	Specifies the canvas border line thickness (in pixels) and should be an integer in the range from 0 to 15.
border_style	Specifies the canvas border line style and should be **bevel** or **drop shadow** or **solid** (or any unique prefix).
margin	Specifies the margin space outside the canvas border (in pixels) and should be an integer in the range from 0 to 31.
margin_height	Specifies the margin space outside the top and bottom sides of the canvas border (in pixels) and should be an integer in the range from 0 to 31.
margin_width	Specifies the margin space outside the left and right sides of the canvas border (in pixels) and should be an integer in the range from 0 to 31.
padding	Specifies the padding space inside the canvas border (in pixels) and should be an integer in the range from 0 to 31.
padding_height	Specifies the padding space inside the top and bottom sides of the canvas border (in pixels) and should be an integer in the range from 0 to 31.
padding_width	Specifies the padding space inside the left and right sides of the canvas border (in pixels) and should be an integer in the range from 0 to 31.

Container Related Traits

min_height	Specifies the minimum height of the canvas (in pixels) and should be an integer in the range from 0 to 99999.

min_width	Specifies the minimum width of the canvas (in pixels) and should be an integer in the range from 0 to 99999.
weight	Specifies the weight of the canvas's size relative to other canvas's in the same container and should be a floating point number in the range from 0.0 to 10.0.

General Chaco Object Related Traits

container	Specifies the object containing this object and should be a Container.
selected	Specifies whether the select markers should be drawn and should be **f** or **false** or **n** or **no** or **off** or **on** or **t** or **true** or **y** or **yes** or 0 or 1.
label	Specifies a brief description of the object and should be of type **str**.
debug	Specifies whether the margin and padding area bounding boxes should be drawn and should be **f** or **false** or **n** or **no** or **off** or **on** or **t** or **true** or **y** or **yes** or 0 or 1.
visible	Specifies whether the canvas object is visible and should be **f** or **false** or **n** or **no** or **off** or **on** or **t** or **true** or **y** or **yes** or 0 or 1.
menu	Specifies the canvas pop-up menu and should be of type **str**.

PlotIndexed

A **PlotIndexed** object visually factors out common **index** axes from a collection of contained **PlotValue** and **PlotCanvas** objects.

In many ways a **PlotIndexed** object behaves like a **PlotGroup** object, laying out its contained objects either vertically or horizontally, depending upon the setting of its **index_axis** trait. The main difference is the visual factoring of the shared **index** axes from all plottable objects it contains.

A **PlotIndexed** object can be added to a **PlotGroup**, and may contain **PlotCanvas** and **PlotValue** objects.

Indexed Related Traits

index_scrollbar	Specifies the scrollbar to use for scrolling the index and should be None.
value_scrollbar	Specifies the scrollbar to use for scrolling the value and should be None.

Plottable Object Related Traits

titles	Specifies None and should be any value.
overlays	Specifies None and should be any value.
values	Specifies None and should be any value.
overlay	Specifies whether the canvas overlays its container and should be **f** or **false** or **n** or **no** or **off** or **on** or **t** or **true** or **y** or **yes** or 0 or 1.
position	Specifies the canvas's position relative to its container and should be **b** or **bl** or **bottom center** or **bottom left** or **bottom right** or **bottom** or **br** or **c** or **center bottom** or **center left** or **center right** or **center top** or **center** or **l** or **lb** or **left bottom** or **left center** or **left top** or **left** or **lt** or **r** or **rb** or **right bottom** or **right center** or **right top** or **right** or **rt** or **t** or **tl** or **top center** or **top left** or **top right** or **top** or **tr**.
index_scale	Specifies the index scaling factor and should be a floating point number in the range from 1.0 to 100.0.
value_scale	Specifies the value scaling factor and should be a floating point number in the range from 1.0 to 100.0.
cursor_type	Specifies the plotting area cursor drawing type and should be **both** or **index** or **none** or **value** (or any unique prefix).
cursor_color	Specifies the cursor line color and should be a number, which in hex is of the form 0xrrggbb, where rr is red, gg is green, and bb is blue, or a sequence of 4 elements (rr, gg, bb, aa), where aa is alpha, with each element a number in the range from 0.0 to 1.0 or any standard color name.

cursor_style	Specifies the cursor line style and should be **dash** or **dot dash** or **dot** or **long dash** or **solid** (or any unique prefix).
cursor_weight	Specifies the cursor line weight (in pixels) and should be an integer in the range from 1 to 9.
index	Specifies None and should be a PlotValue or None.
index_axis	Specifies the axis the index axis is aligned with and should be **x** or **y**.
axis	Specifies None and should be a PlotAxis.
axis_index	Specifies None and should be a PlotAxis or None.

Frame Related Traits

bg_image	Specifies the background image and should be an Image or an image file name (e.g. a **.gif**, **.png** or **.jpeg** file) or **.**
bg_color	Specifies the canvas background color and should be a number, which in hex is of the form 0xrrggbb, where rr is red, gg is green, and bb is blue, or a sequence of 4 elements (rr, gg, bb, aa), where aa is alpha, with each element a number in the range from 0.0 to 1.0 or None or any standard color name.
border_color	Specifies the canvas border line color and should be a number, which in hex is of the form 0xrrggbb, where rr is red, gg is green, and bb is blue, or a sequence of 4 elements (rr, gg, bb, aa), where aa is alpha, with each element a number in the range from 0.0 to 1.0 or any standard color name.
border_size	Specifies the canvas border line thickness (in pixels) and should be an integer in the range from 0 to 15.
border_style	Specifies the canvas border line style and should be **bevel** or **drop shadow** or **solid** (or any unique prefix).
margin	Specifies the margin space outside the canvas border (in pixels) and should be an integer in the range from 0 to 31.
margin_height	Specifies the margin space outside the top and bottom sides of the canvas border (in pixels) and should be an integer in the range from 0 to 31.
margin_width	Specifies the margin space outside the left and right sides of the canvas border (in pixels) and should be an integer in the range from 0 to 31.
padding	Specifies the padding space inside the canvas border (in pixels) and should be an integer in the range from 0 to 31.
padding_height	Specifies the padding space inside the top and bottom sides of the canvas border (in pixels) and should be an integer in the range from 0 to 31.
padding_width	Specifies the padding space inside the left and right sides of the canvas border (in pixels) and should be an integer in the range from 0 to 31.

Container Related Traits

min_height	Specifies the minimum height of the canvas (in pixels) and should be an integer in the range from 0 to 99999.
min_width	Specifies the minimum width of the canvas (in pixels) and should be an integer in the range from 0 to 99999.
weight	Specifies the weight of the canvas's size relative to other canvas's in the same container and should be a floating point number in the range from 0.0 to 10.0.

General Chaco Object Related Traits

container	Specifies the object containing this object and should be a Container.
selected	Specifies whether the select markers should be drawn and should be **f** or **false** or **n** or **no** or **off** or **on** or **t** or **true** or **y** or **yes** or 0 or 1.
label	Specifies a brief description of the object and should be of type **str**.
debug	Specifies whether the margin and padding area bounding boxes should be drawn and should be **f** or **false** or **n** or **no** or **off** or **on** or **t** or **true** or **y** or **yes** or 0 or 1.
visible	Specifies whether the canvas object is visible and should be **f** or **false** or **n** or **no** or **off** or **on** or **t** or **true** or **y** or **yes** or 0 or 1.
menu	Specifies the canvas pop-up menu and should be of type **str**.

PlotGroup

A **PlotGroup** object lays out its contained objects either horizontally or vertically, depending upon the value of its **orientation** trait.

A **PlotGroup** can contain **PlotValue**, **PlotCanvas** and **PlotIndexed** objects, as well as other **PlotGroup** objects.

Group Related Traits

orientation	Specifies the layout direction for contained objects and should be **horizontal** or **vertical** (or any unique prefix).

Plottable Object Related Traits

titles	Specifies None and should be any value.
overlays	Specifies None and should be any value.
values	Specifies None and should be any value.
overlay	Specifies whether the canvas overlays its container and should be **f** or **false** or **n** or **no** or **off** or **on** or **t** or **true** or **y** or **yes** or 0 or 1.
position	Specifies the canvas's position relative to its container and should be **b** or **bl** or **bottom center** or **bottom left** or **bottom right** or **bottom** or **br** or **c** or **center bottom** or **center left** or **center right** or **center top** or **center** or **l** or **lb** or **left bottom** or **left center** or **left top** or **left** or **lt** or **r** or **rb** or **right bottom** or **right center** or **right top** or **right** or **rt** or **t** or **tl** or **top center** or **top left** or **top right** or **top** or **tr**.
index_scale	Specifies the index scaling factor and should be a floating point number in the range from 1.0 to 100.0.
value_scale	Specifies the value scaling factor and should be a floating point number in the range from 1.0 to 100.0.
cursor_type	Specifies the plotting area cursor drawing type and should be **both** or **index** or **none** or **value** (or any unique prefix).
cursor_color	Specifies the cursor line color and should be a number, which in hex is of the form 0xrrggbb, where rr is red, gg is green, and bb is blue, or a sequence of 4 elements (rr, gg, bb, aa), where aa is alpha, with each element a number in the range from 0.0 to 1.0 or any standard color name.
cursor_style	Specifies the cursor line style and should be **dash** or **dot dash** or **dot** or **long dash** or **solid** (or any unique prefix).
cursor_weight	Specifies the cursor line weight (in pixels) and should be an integer in the range from 1 to 9.
index	Specifies None and should be a PlotValue or None.
index_axis	Specifies the axis the index axis is aligned with and should be **x** or **y**.

axis	Specifies None and should be a PlotAxis.
axis_index	Specifies None and should be a PlotAxis or None.

Frame Related Traits

bg_image	Specifies the background image and should be an Image or an image file name (e.g. a **.gif**, **.png** or **.jpeg** file) or **.**
bg_color	Specifies the canvas background color and should be a number, which in hex is of the form 0xrrggbb, where rr is red, gg is green, and bb is blue, or a sequence of 4 elements (rr, gg, bb, aa), where aa is alpha, with each element a number in the range from 0.0 to 1.0 or None or any standard color name.
border_color	Specifies the canvas border line color and should be a number, which in hex is of the form 0xrrggbb, where rr is red, gg is green, and bb is blue, or a sequence of 4 elements (rr, gg, bb, aa), where aa is alpha, with each element a number in the range from 0.0 to 1.0 or any standard color name.
border_size	Specifies the canvas border line thickness (in pixels) and should be an integer in the range from 0 to 15.
border_style	Specifies the canvas border line style and should be **bevel** or **drop shadow** or **solid** (or any unique prefix).
margin	Specifies the margin space outside the canvas border (in pixels) and should be an integer in the range from 0 to 31.
margin_height	Specifies the margin space outside the top and bottom sides of the canvas border (in pixels) and should be an integer in the range from 0 to 31.
margin_width	Specifies the margin space outside the left and right sides of the canvas border (in pixels) and should be an integer in the range from 0 to 31.
padding	Specifies the padding space inside the canvas border (in pixels) and should be an integer in the range from 0 to 31.
padding_height	Specifies the padding space inside the top and bottom sides of the canvas border (in pixels) and should be an integer in the range from 0 to 31.
padding_width	Specifies the padding space inside the left and right sides of the canvas border (in pixels) and should be an integer in the range from 0 to 31.

Container Related Traits

min_height	Specifies the minimum height of the canvas (in pixels) and should be an integer in the range from 0 to 99999.
min_width	Specifies the minimum width of the canvas (in pixels) and should be an integer in the range from 0 to 99999.
weight	Specifies the weight of the canvas's size relative to other canvas's in the same container and should be a floating point number in the range from 0.0 to 10.0.

General Chaco Object Related Traits

container	Specifies the object containing this object and should be a Container.
selected	Specifies whether the select markers should be drawn and should be **f** or **false** or **n** or **no** or **off** or **on** or **t** or **true** or **y** or **yes** or 0 or 1.
label	Specifies a brief description of the object and should be of type **str**.
debug	Specifies whether the margin and padding area bounding boxes should be drawn and should be **f** or **false** or **n** or **no** or **off** or **on** or **t** or **true** or **y** or **yes** or 0 or 1.
visible	Specifies whether the canvas object is visible and should be **f** or **false** or **n** or **no** or **off** or **on** or **t** or **true** or **y** or **yes** or 0 or 1.
menu	Specifies the canvas pop-up menu and should be of type **str**.

PlotWindow

A **PlotWindow** object is a GUI toolkit specific control that displays a Chaco container object and its contents in the display area assigned to the **PlotWindow**. There are implementations of **PlotWindow** for both the wxPython and Tkinter toolkits.

A **PlotWindow** object also handles user interface events such as mouse click and drag operations, and assists in mapping them into Chaco plotting object specific events. **PlotWindow** objects also create and manage the kiva graphics context used to render Chaco objects into a raster form suitable for viewing on a display screen.

The Chaco object displayed by a **PlotWindow** object is specified by assigning it as the value of the **PlotWindow canvas** trait. Any Chaco container object (i.e. **PlotGroup**, **PlotIndexed**, **PlotCanvas**, or **PlotValue**) can be assigned to the **canvas** trait.

Window Related Traits

canvas	Specifies None and should be a **PlotGroup** or a **PlotIndexed** or a **PlotCanvas** or **None**.
bg_color	Specifies None and should be a number, which in hex is of the form 0xrrggbb, where rr is red, gg is green, and bb is blue, or a sequence of 4 elements (rr, gg, bb, aa), where aa is alpha, with each element a number in the range from 0.0 to 1.0 or any **standard color name**.
right_click	Specifies None and should be **ae** or **all edit** or **all menu** or **am** or **edit** or **ignore** or **menu** or **te** or **tm** or **top edit** or **top menu** (or any unique prefix).

DefaultContainer

A **DefaultContainer** object defines a collection of *inheritable* trait values that objects contained within a Chaco container use as their corresponding trait's default values.

In general, the trait names defined by a **DefaultContainer** are related to the correponding contained object's trait names, but with the addition of a qualify prefix to indicate the type of contained object the trait applies to. For example, the **DefaultContainer title_bg_color** trait corresponds to the **PlotTitle bg_color** trait.

Any Chaco object that behaves as a *container* (i.e. **PlotGroup**, **PlotIndexed**, **PlotCanvas**, **PlotValue**) has all of the traits defined here. They are not listed again under each Chaco container object.

Container Related Traits

canvas_bg_color	Specifies the canvas background color and should be a number, which in hex is of the form 0xrrggbb, where rr is red, gg is green, and bb is blue, or a sequence of 4 elements (rr, gg, bb, aa), where aa is alpha, with each element a number in the range from 0.0 to 1.0 or None or any standard color name.
canvas_border_color	Specifies the canvas border line color and should be a number, which in hex is of the form 0xrrggbb, where rr is red, gg is green, and bb is blue, or a sequence of 4 elements (rr, gg, bb, aa), where aa is alpha, with each element a number in the range from 0.0 to 1.0 or any standard color name.
canvas_border_size	Specifies the canvas border line thickness (in pixels) and should be an integer in the range from 0 to 15.
canvas_border_style	Specifies the canvas border line style and should be **bevel** or **drop shadow** or **solid** (or any unique prefix).
canvas_margin	Specifies the margin space outside the canvas border (in pixels) and should be an integer in the range from 0 to 31.
canvas_margin_height	Specifies the margin space outside the top and bottom sides of the canvas border (in pixels) and should be an integer in the range from 0 to 31.
canvas_margin_width	Specifies the margin space outside the left and right sides of the canvas border (in pixels) and should be an integer in the range from 0 to 31.
canvas_padding	Specifies the padding space inside the canvas border (in pixels) and should be an integer in the range from 0 to 31.
canvas_padding_height	Specifies the padding space inside the top and bottom sides of the canvas border (in pixels) and should be an integer in the range from 0 to 31.

canvas_padding_width	Specifies the padding space inside the left and right sides of the canvas border (in pixels) and should be an integer in the range from 0 to 31.
canvas_position	Specifies the canvas's position relative to its container and should be **b** or **bl** or **bottom center** or **bottom left** or **bottom right** or **bottom** or **br** or **c** or **center bottom** or **center left** or **center right** or **center top** or **center** or **l** or **lb** or **left bottom** or **left center** or **left top** or **left** or **lt** or **r** or **rb** or **right bottom** or **right center** or **right top** or **right** or **rt** or **t** or **tl** or **top center** or **top left** or **top right** or **top** or **tr**.
canvas_overlay	Specifies whether the canvas overlays its container and should be **f** or **false** or **n** or **no** or **off** or **on** or **t** or **true** or **y** or **yes** or 0 or 1.
canvas_visible	Specifies whether the canvas object is visible and should be **f** or **false** or **n** or **no** or **off** or **on** or **t** or **true** or **y** or **yes** or 0 or 1.
canvas_menu	Specifies the canvas pop-up menu and should be of type **str**.
canvas_select_type	Specifies the type of range the user can drag select and should be **index** or **none** or **value** (or any unique prefix).
canvas_select_notify	Specifies the routine to be called when a drag select operation is complete and should be of type **function** or of type **instance method** or None.
canvas_index_scale	Specifies the index scaling factor and should be a floating point number in the range from 1.0 to 100.0.
canvas_value_scale	Specifies the value scaling factor and should be a floating point number in the range from 1.0 to 100.0.

Panel Related Traits

panel_visible	Specifies whether the panel object is visible and should be **f** or **false** or **n** or **no** or **off** or **on** or **t** or **true** or **y** or **yes** or 0 or 1.
panel_bg_color	Specifies the panel background color and should be a number, which in hex is of the form 0xrrggbb, where rr is red, gg is green, and bb is blue, or a sequence of 4 elements (rr, gg, bb, aa), where aa is alpha, with each element a number in the range from 0.0 to 1.0 or None or any standard color name.
panel_border_color	Specifies the panel border line color and should be a number, which in hex is of the form 0xrrggbb, where rr is red, gg is green, and bb is blue, or a sequence of 4 elements (rr, gg, bb, aa), where aa is alpha, with each element a number in the range from 0.0 to 1.0 or any standard color name.
panel_border_size	Specifies the panel border line thickness (in pixels) and should be an integer in the range from 0 to 15.
panel_border_style	Specifies the panel border line style and should be **bevel** or **drop shadow** or **solid** (or any unique prefix).

panel_margin	Specifies the margin space outside the panel border (in pixels) and should be an integer in the range from 0 to 31.
panel_margin_height	Specifies the margin space outside the top and bottom sides of the panel border (in pixels) and should be an integer in the range from 0 to 31.
panel_margin_width	Specifies the margin space outside the left and right sides of the panel border (in pixels) and should be an integer in the range from 0 to 31.
panel_padding	Specifies the padding space inside the panel border (in pixels) and should be an integer in the range from 0 to 31.
panel_padding_height	Specifies the padding space inside the top and bottom sides of the panel border (in pixels) and should be an integer in the range from 0 to 31.
panel_padding_width	Specifies the padding space inside the left and right sides of the panel border (in pixels) and should be an integer in the range from 0 to 31.
panel_menu	Specifies the panel pop-up menu and should be of type **str**.
panel_weight	Specifies None and should be **stretch**.

Overlay Related Traits

overlay_color	Specifies the overlay text color and should be a number, which in hex is of the form 0xrrggbb, where rr is red, gg is green, and bb is blue, or a sequence of 4 elements (rr, gg, bb, aa), where aa is alpha, with each element a number in the range from 0.0 to 1.0 or any standard color name.
overlay_legend_color	Specifies the overlay legend color and should be a number, which in hex is of the form 0xrrggbb, where rr is red, gg is green, and bb is blue, or a sequence of 4 elements (rr, gg, bb, aa), where aa is alpha, with each element a number in the range from 0.0 to 1.0 or None or any standard color name.
overlay_font	Specifies the overlay text font and should be a Font or a string describing a font (e.g. **12 pt bold italic swiss family Arial** or **default 12**).
overlay_rotate	Specifies the overlay rotation (in degrees) and should be 0 or 90.
overlay_visible	Specifies whether the overlay is visible and should be **f** or **false** or **n** or **no** or **off** or **on** or **t** or **true** or **y** or **yes** or 0 or 1.
overlay_bg_color	Specifies the overlay background color and should be a number, which in hex is of the form 0xrrggbb, where rr is red, gg is green, and bb is blue, or a sequence of 4 elements (rr, gg, bb, aa), where aa is alpha, with each element a number in the range from 0.0 to 1.0 or None or any standard color name.
overlay_border_color	Specifies the overlay border line color and should be a number, which in hex is of the form 0xrrggbb, where rr is red, gg is green,

	and bb is blue, or a sequence of 4 elements (rr, gg, bb, aa), where aa is alpha, with each element a number in the range from 0.0 to 1.0 or any standard color name.
overlay_border_size	Specifies the overlay border line thickness (in pixels) and should be an integer in the range from 0 to 15.
overlay_border_style	Specifies the overlay border line style and should be **bevel** or **drop shadow** or **solid** (or any unique prefix).
overlay_margin	Specifies the margin space outside the overlay border (in pixels) and should be an integer in the range from 0 to 31.
overlay_margin_height	Specifies the margin space outside the top and bottom sides of the overlay border (in pixels) and should be an integer in the range from 0 to 31.
overlay_margin_width	Specifies the margin space outside the left and right sides of the overlay border (in pixels) and should be an integer in the range from 0 to 31.
overlay_padding	Specifies the padding space inside the overlay border (in pixels) and should be an integer in the range from 0 to 31.
overlay_padding_height	Specifies the padding space inside the top and bottom sides of the overlay border (in pixels) and should be an integer in the range from 0 to 31.
overlay_padding_width	Specifies the padding space inside the left and right sides of the overlay border (in pixels) and should be an integer in the range from 0 to 31.
overlay_position	Specifies the overlay's position relative to its container and should be **b** or **bl** or **bottom center** or **bottom left** or **bottom right** or **bottom** or **br** or **c** or **center bottom** or **center left** or **center right** or **center top** or **center** or **l** or **lb** or **left bottom** or **left center** or **left top** or **left** or **lt** or **r** or **rb** or **right bottom** or **right center** or **right top** or **right** or **rt** or **t** or **tl** or **top center** or **top left** or **top right** or **top** or **tr**.
overlay_alignment	Specifies the alignment of the overlay text within its border and should be **bottom** or **center** or **left** or **right** or **top** (or any unique prefix).
overlay_menu	Specifies the overlay pop-up menu and should be of type **str**.
overlay_value	Specifies the overlay value coordinate and should be of type **float** or None.
overlay_index	Specifies the overlay index coordinate and should be of type **float** or None.
overlay_line_style	Specifies the overlay line style and should be **dash** or **dot dash** or **dot** or **long dash** or **solid** (or any unique prefix).
overlay_line_color	Specifies the overlay line color and should be a number, which in hex is of the form 0xrrggbb, where rr is red, gg is green, and bb is blue, or a sequence of 4 elements (rr, gg, bb, aa), where aa is alpha, with each element a number in the range from 0.0 to 1.0 or None or any standard color name.

overlay_line_weight	Specifies the overlay line weight (in pixels) and should be an integer in the range from 1 to 9.

Title Related Traits

title_color	Specifies the title text color and should be a number, which in hex is of the form 0xrrggbb, where rr is red, gg is green, and bb is blue, or a sequence of 4 elements (rr, gg, bb, aa), where aa is alpha, with each element a number in the range from 0.0 to 1.0 or any standard color name.
title_legend_color	Specifies the title legend color and should be a number, which in hex is of the form 0xrrggbb, where rr is red, gg is green, and bb is blue, or a sequence of 4 elements (rr, gg, bb, aa), where aa is alpha, with each element a number in the range from 0.0 to 1.0 or None or any standard color name.
title_font	Specifies the title text font and should be a Font or a string describing a font (e.g. **12 pt bold italic swiss family Arial** or **default 12**).
title_rotate	Specifies the title rotation (in degrees) and should be 0 or 90.
title_visible	Specifies whether the title is visible and should be **f** or **false** or **n** or **no** or **off** or **on** or **t** or **true** or **y** or **yes** or 0 or 1.
title_bg_color	Specifies the title background color and should be a number, which in hex is of the form 0xrrggbb, where rr is red, gg is green, and bb is blue, or a sequence of 4 elements (rr, gg, bb, aa), where aa is alpha, with each element a number in the range from 0.0 to 1.0 or None or any standard color name.
title_border_color	Specifies the title border line color and should be a number, which in hex is of the form 0xrrggbb, where rr is red, gg is green, and bb is blue, or a sequence of 4 elements (rr, gg, bb, aa), where aa is alpha, with each element a number in the range from 0.0 to 1.0 or any standard color name.
title_border_size	Specifies the title border line thickness (in pixels) and should be an integer in the range from 0 to 15.
title_border_style	Specifies the title border line style and should be **bevel** or **drop shadow** or **solid** (or any unique prefix).
title_margin	Specifies the margin space outside the title border (in pixels) and should be an integer in the range from 0 to 31.
title_margin_height	Specifies the margin space outside the top and bottom sides of the title border (in pixels) and should be an integer in the range from 0 to 31.
title_margin_width	Specifies the margin space outside the left and right sides of the title border (in pixels) and should be an integer in the range from 0 to 31.
title_padding	Specifies the padding space inside the title border (in pixels) and should be an integer in the range from 0 to 31.
title_padding_height	Specifies the padding space inside the top and bottom sides of the title border (in pixels) and should be an integer in the range from 0 to 31.

title_padding_width	Specifies the padding space inside the left and right sides of the title border (in pixels) and should be an integer in the range from 0 to 31.
title_position	Specifies the side of the canvas the title appears on and should be **bottom** or **left** or **right** or **top** (or any unique prefix).
title_alignment	Specifies the alignment of the title text within its border and should be **bottom** or **center** or **left** or **right** or **top** (or any unique prefix).
title_menu	Specifies the title pop-up menu and should be of type **str**.

Plot (i.e. PlotValue) Related Traits

plot_type	Specifies the type of plot drawn and should be a comma separated plot type string (e.g. **line,scatter**) or plot type list (e.g. ['line,'scatter']).
plot_origin	Specifies the corner of the plotting area used as the axis origin and should be **bl** or **bottom left** or **bottom right** or **br** or **lb** or **left bottom** or **left top** or **lt** or **rb** or **right bottom** or **right top** or **rt** or **tl** or **top left** or **top right** or **tr**.
plot_bg_color	Specifies the plot background color and should be a number, which in hex is of the form 0xrrggbb, where rr is red, gg is green, and bb is blue, or a sequence of 4 elements (rr, gg, bb, aa), where aa is alpha, with each element a number in the range from 0.0 to 1.0 or None or any standard color name.
plot_border_color	Specifies the plot border line color and should be a number, which in hex is of the form 0xrrggbb, where rr is red, gg is green, and bb is blue, or a sequence of 4 elements (rr, gg, bb, aa), where aa is alpha, with each element a number in the range from 0.0 to 1.0 or any standard color name.
plot_border_size	Specifies the plot border line thickness (in pixels) and should be an integer in the range from 0 to 15.
plot_border_style	Specifies the plot border line style and should be **bevel** or **drop shadow** or **solid** (or any unique prefix).
plot_line_style	Specifies the plot line style and should be **dash** or **dot dash** or **dot** or **long dash** or **solid** (or any unique prefix).
plot_line_color	Specifies the plot line color and should be a number, which in hex is of the form 0xrrggbb, where rr is red, gg is green, and bb is blue, or a sequence of 4 elements (rr, gg, bb, aa), where aa is alpha, with each element a number in the range from 0.0 to 1.0 or None or any standard color name.
plot_line_weight	Specifies the plot line weight (in pixels) and should be an integer in the range from 1 to 9.
plot_outline_color	Specifies the plot marker outline color and should be a number, which in hex is of the form 0xrrggbb, where rr is red, gg is green, and bb is blue, or a sequence of 4 elements (rr, gg, bb, aa), where aa is alpha, with each element a number in the range from 0.0 to 1.0 or any standard color name.

plot_outline_weight	Specifies the plot marker outline weight (in pixels) and should be an integer in the range from 0 to 9.
plot_fill_color	Specifies the plot marker fill color and should be a number, which in hex is of the form 0xrrggbb, where rr is red, gg is green, and bb is blue, or a sequence of 4 elements (rr, gg, bb, aa), where aa is alpha, with each element a number in the range from 0.0 to 1.0 or any standard color name.
plot_fill_style	Specifies the plot marker fill style and should be **back hatch** or **cross hatch** or **diag hatch** or **forward hatch** or **horz hatch** or **solid** or **stipple** or **transparent** or **vert hatch** (or any unique prefix).
plot_size	Specifies the plot marker size and should be **large** or **medium** or **small** (or any unique prefix).
plot_symbol	Specifies the plot marker symbol type and should be **circle** or **square** or **diamond** or **triangle** or **down triangle** or **cross** or **plus** (or any unique prefix).
plot_bar_width	Specifies the width of a plot bar as a percentage of the available space and should be a floating point number in the range from 10.0 to 150.0.
plot_bin_width	Specifies the width of a group of plot bars as a percentage of the available space and should be a floating point number in the range from 50.0 to 100.0.
plot_min_height	Specifies the minimum height of the plot (in pixels) and should be an integer in the range from 0 to 99999.
plot_min_width	Specifies the minimum width of the plot (in pixels) and should be an integer in the range from 0 to 99999.
plot_padding	Specifies the padding space inside the plot border (in pixels) and should be an integer in the range from 0 to 31.
plot_padding_height	Specifies the padding space inside the top and bottom sides of the plot (in pixels) and should be an integer in the range from 0 to 31.
plot_padding_width	Specifies the padding space inside the left and right sides of the plot border (in pixels) and should be an integer in the range from 0 to 31.

Axis Related Traits

axis_visible	Specifies whether the axis is visible and should be **f** or **false** or **n** or **no** or **off** or **on** or **t** or **true** or **y** or **yes** or 0 or 1.
axis_scale	Specifies the scaling mode of the axis and should be **linear** or **log** (or any unique prefix).
axis_position	Specifies the sides of the plot the axis appears on and should be **all** or **both** or **bottom** or **left** or **right** or **top** (or any unique prefix).
axis_menu	Specifies the axis pop-up menu and should be of type **str**.
axis_line_weight	Specifies the axis line weight (in pixels) and should be an integer in the range from 1 to 9.

axis_line_color	Specifies the axis line color and should be a number, which in hex is of the form 0xrrggbb, where rr is red, gg is green, and bb is blue, or a sequence of 4 elements (rr, gg, bb, aa), where aa is alpha, with each element a number in the range from 0.0 to 1.0 or any standard color name.
axis_line_style	Specifies the axis line style and should be **dash** or **dot dash** or **dot** or **long dash** or **solid** (or any unique prefix).
axis_label_font	Specifies the axis label font and should be a Font or a string describing a font (e.g. **12 pt bold italic swiss family Arial** or **default 12**).
axis_label_color	Specifies the axis label text color and should be a number, which in hex is of the form 0xrrggbb, where rr is red, gg is green, and bb is blue, or a sequence of 4 elements (rr, gg, bb, aa), where aa is alpha, with each element a number in the range from 0.0 to 1.0 or any standard color name.
axis_label_style	Specifies the location of labels along the axis and should be an integer in the range from 1 to 10 or **all** or **edge** or **end** or **none** (or any unique prefix).
axis_label_offset	Specifies the offset of axis labels from the tick marks and should be an integer in the range from 0 to 31.
axis_label_format	Specifies the format to apply when creating axis labels and should be a standard Python conversion string (e.g. **%%.2f**) or a function which converts a number to a string.
axis_tick_visible	Specifies whether the axis tick marks are visible and should be **f** or **false** or **n** or **no** or **off** or **on** or **t** or **true** or **y** or **yes** or 0 or 1.
axis_tick_weight	Specifies the axis tick mark line weight (in pixels) and should be an integer in the range from 1 to 9.
axis_tick_color	Specifies the axis tick mark color and should be a number, which in hex is of the form 0xrrggbb, where rr is red, gg is green, and bb is blue, or a sequence of 4 elements (rr, gg, bb, aa), where aa is alpha, with each element a number in the range from 0.0 to 1.0 or any standard color name.
axis_tick_style	Specifies the axis tick mark line style and should be **dash** or **dot dash** or **dot** or **long dash** or **solid** (or any unique prefix).
axis_tick_in	Specifies the length of axis ticks into the plot (in pixels) and should be an integer in the range from 0 to 31.
axis_tick_out	Specifies the length of axis ticks out from the plot (in pixels) and should be an integer in the range from 0 to 31.
axis_tick_interval	Specifies the spacing of axis tick marks and should be **auto** (or any unique prefix) or a floating point number in the range from 0.0 to 1e+308 or an integer in the range from -100 to -1.
axis_grid_visible	Specifies whether the axis grid lines are visible and should be **f** or **false** or **n** or **no** or **off** or **on** or **t** or **true** or **y** or **yes** or 0 or 1.
axis_grid_weight	Specifies the axis grid line weight (in pixels) and should be an integer in the range from 1 to 9.

axis_grid_color	Specifies the axis grid line color and should be a number, which in hex is of the form 0xrrggbb, where rr is red, gg is green, and bb is blue, or a sequence of 4 elements (rr, gg, bb, aa), where aa is alpha, with each element a number in the range from 0.0 to 1.0 or any standard color name.
axis_grid_style	Specifies the axis grid line style and should be **dash** or **dot dash** or **dot** or **long dash** or **solid** (or any unique prefix).
axis_margin	Specifies the margin space around the outside of the axis (in pixels) and should be an integer in the range from 0 to 31.
axis_margin_height	Specifies the margin space outside the top and bottom sides of the axis (in pixels) and should be an integer in the range from 0 to 31.
axis_margin_width	Specifies the margin space outside the left and right sides of the axis (in pixels) and should be an integer in the range from 0 to 31.
axis_bound_low	Specifies the lower axis bound and should be **auto** or **fit** (or any unique prefix) or of type **float**.
axis_bound_high	Specifies the upper axis bound and should be **auto** or **fit** (or any unique prefix) or of type **float**.
axis_zoomable	Specifies whether the axis can be drag zoomed by the user and should be **f** or **false** or **n** or **no** or **off** or **on** or **t** or **true** or **y** or **yes** or 0 or 1.

Container Layout Related Traits

min_height	Specifies the minimum height of the canvas (in pixels) and should be an integer in the range from 0 to 99999.
min_width	Specifies the minimum width of the canvas (in pixels) and should be an integer in the range from 0 to 99999.
weight	Specifies the weight of the canvas's size relative to other canvas's in the same container and should be a floating point number in the range from 0.0 to 10.0.

Cursor Related Traits

cursor_type	Specifies the plotting area cursor drawing type and should be **both** or **index** or **none** or **value** (or any unique prefix).
cursor_style	Specifies the cursor line style and should be **dash** or **dot dash** or **dot** or **long dash** or **solid** (or any unique prefix).
cursor_color	Specifies the cursor line color and should be a number, which in hex is of the form 0xrrggbb, where rr is red, gg is green, and bb is blue, or a sequence of 4 elements (rr, gg, bb, aa), where aa is alpha, with each element a number in the range from 0.0 to 1.0 or any standard color name.
cursor_weight	Specifies the cursor line weight (in pixels) and should be an integer in the range from 1 to 9.

Miscellaneous Other Traits

index_axis	Specifies the axis the index axis is aligned with and should be **x** or **y**.
debug	Specifies whether the margin and padding area bounding boxes should be drawn and should be **f** or **false** or **n** or **no** or **off** or **on** or **t** or **true** or **y** or **yes** or 0 or 1.
axis	Specifies None and should be a **PlotAxis**.
axis_index	Specifies None and should be a **PlotAxis** or None.
index	Specifies None and should be a **PlotValue** or None.

Standard Color Names

The following are the standard color names that can be used with any Chaco color related trait (i.e. any trait having a name ending in **color**):

- **aquamarine**
- **black**
- **blue violet**
- **blue**
- **brown**
- **cadet blue**
- **coral**
- **cornflower blue**
- **cyan**
- **dark green**
- **dark grey**
- **dark olive green**
- **dark orchid**
- **dark slate blue**
- **dark slate grey**
- **dark turquoise**
- **dim grey**
- **firebrick**
- **forest green**
- **gold**
- **goldenrod**
- **green yellow**
- **green**
- **grey**
- **indian red**
- **khaki**
- **light blue**
- **light grey**
- **light steel**
- **lime green**
- **magenta**
- **maroon**
- **medium aquamarine**
- **medium blue**
- **medium forest green**
- **medium goldenrod**
- **medium orchid**
- **medium sea green**
- **medium slate blue**
- **medium spring green**
- **medium turquoise**
- **medium violet red**
- **midnight blue**
- **navy**
- **orange red**
- **orange**
- **orchid**
- **pale green**
- **pink**
- **plum**
- **purple**
- **red**
- **salmon**
- **sea green**
- **sienna**
- **sky blue**
- **slate blue**
- **spring green**
- **steel blue**
- **tan**
- **thistle**
- **transparent**
- **turquoise**
- **violet red**
- **violet**
- **wheat**
- **white**
- **yellow green**
- **yellow**

The PlotWindow Class

The **PlotWindow** class acts as a bridge between the Chaco plotting objects we've already discussed and a particular GUI toolkit, such as wxPython or Tkinter. There are multiple implementations of the **PlotWindow** class, one for each supported GUI toolkit.

A **PlotWindow** instance is a normal wxPython or Tkinter *widget* that can be imbedded like any other widget in a wxPython or Tkinter window or dialog. Each **PlotWindow** object renders on the user's display a single Chaco *container* object, such as a **PlotGroup**, that is associated with it through the **PlotWindow's canvas** trait.

Among the functions performed by a **PlotWindow** object are:

- The creation and management of the **kiva** graphics context used to render the associated Chaco container object on the user's display.
- Mouse and keyboard event handling.
- The instantiation of trait editor dialogs.
- The creation of the contextual pop-up menus associated with various Chaco plotting objects.
- The creation and management of scrollbars used to scroll the contents of Chaco container objects whose size exceeds the available display area.
- The drawing of various *rubberbanding* effects used to implement axis zooming and value or index selection.

Because a **PlotWindow** is GUI toolkit specific, the constructor is different for each toolkit.

For wxPython, the constructor has the form:

**PlotWindow(parent, wid = -1, pos = wx.wxPyDefaultPosition,
size = wx.wxPyDefaultSize, [trait = value, trait = value, …])**
Creates an wxPython-based **PlotWindow** object. The **parent**, **wid**, **pos**, and **size** arguments are standard wxPython widget constructor arguments. Refer to the wxPython documentation for more information about their meaning. In addition, any number of **PlotWindow** object traits can also be set by specifying the appropriate trait and value (i.e. **trait = value**) pairs.

For Tkinter, the constructor has the form:

PlotWindow(parent, [trait = value, trait = value, …])
Creates a Tkinter-based **PlotWindow** object. The **parent** argument is a standard Tkinter widget constructor argument. Refer to the Tkinter documentation for more information about its meaning. In addition, any number of **PlotWindow** object traits can also be set by specifying the appropriate trait and value (i.e. **trait = value**) pairs.

Each **PlotWindow** implementation subclasses from a particular GUI toolkit specific base class. In the case of wxPython, the base class is **wxScrolledWindow**. In the case of Tkinter, the base class is **Widget**. As such, an application is free to use any of the methods defined by those base classes.

Additionally, each **PlotWindow** implementation also defines the following Chaco specific methods:

add([object, object, …])

Adds each of the specified **objects** to the Chaco container object assigned to the **PlotWindow’s canvas** trait. If no container object has been assigned to the **canvas** trait yet, a **PlotGroup** object is created and assigned to the **canvas** trait before the specified objects are added.

remove([object, object, …])

Removes each of the specified **objects** from the Chaco container object assigned to the **PlotWindow’s canvas** trait. If no container object has been assigned to the canvas trait, the request is ignored.

If no **objects** are specified, then the Chaco container object currently assigned to the **canvas** trait is disassociated from the **PlotWindow** by assigning **None** to the **canvas** trait.

select(object)

Selects the specified Chaco **object**. Any previously selected object is automatically unselected. If **object** is **None**, then after unselecting any previously selected object, no object is selected.

If a trait editor dialog is active, the newly selected object is automatically selected in the editor dialog.

objects_at(x, y = None)

Returns a list of all Chaco objects that contain a specified point. The point may be given as an (**x**,**y**) value specified in window coordinates. Alternatively, **y** may be omitted or **None**, in which case the **x** argument is assumed to be a GUI toolkit specific *event* object from which a window coordinate can be extracted. This form is useful when calling the method from a mouse related event handler called with an *event* object.

The list returned may be empty or contain one or more objects. More than one object is possible since objects can overlap.

The objects are ordered such that the first object in the list is *closest* to the user, and the last is *furthest* away.

Note that **PlotValue** objects contained within a **PlotCanvas** object are never returned by this method. Only the containing **PlotCanvas** object will be returned (if it contains the specified point). Use the **values_at** method to locate **PlotValues** that are near a specified point.

values_at(x, y = None)

Returns a list of all **PlotValue** objects whose visual representation intersects (or is within a small delta of) a specified point. The point may be given as an (**x**,**y**) value specified in window coordinates. Alternatively, **y** may be omitted or **None**, in which case the **x** argument is assumed to be a GUI toolkit specific *event* object from which a window coordinate can be extracted. This form is useful when calling the method from a mouse related event handler called with an *event* object.

The list returned may be empty or contain one or more objects. More than one object is possible since plots can overlap.

update()

Schedules a complete update of the **PlotWindow’s** contents. Note that the actual update of the display will occur sometime after the call to the method, once control has returned to the application’s event dispatch loop.

draw()

Schedules a redraw of the **PlotWindow’s** contents. This is similar to **update**, but does not recalculate sizes or perform a new object layout. As a result, it is somewhat faster than doing an **update**.

if_draw()

Schedules a redraw of the **PlotWindow’s** contents if necessary. In other words, the contents of the **PlotWindow** will only be redrawn if it has been marked as needing it.

on_left_down(event)

This method is called when the user presses the left mouse button while the pointer is over the **PlotWindow** object. **Event** is a GUI toolkit specific object providing information about the event. Refer to the appropriate GUI toolkit documentation (either for wxPython or Tkinter) for more information about the **event** object.

The result returned indicates whether the **PlotWindow** object should perform normal left mouse button down processing or not. Normally processing initiates an axis zoom and index or value select operation if the pointer is over a Chaco object that allows such an operation. If the result returned is 0, this processing is not performed; otherwise it is.

This method is normally overridden by an application specific subclass of **PlotWindow**. The default implementation simply returns 1, to allow normal event processing to be performed.

on_left_up(event)

This method is called when the user releases the left mouse button after having previously pressed it while the pointer was over the **PlotWindow** object. Refer to the **on_left_down** method for more information about **event.** The result returned has no effect on any additional event processing performed.

on_left_dclick(event)

This method is called when the user double-clicks the left mouse button over the **PlotWindow** object. Refer to the **on_left_down** method for more information about **event** and the result returned.

The normal event processing that occurs when the left mouse button is double-clicked is to *unzoom* any axis that has been previously *zoomed*. This processing does not occur if 0 is returned as the result. The default implementation returns 1.

on_mouse_move(event)

This method is called when the user moves the mouse over the **PlotWindow** object. Refer to the **on_left_down** method for more information about **event** and the result returned.

The normal event processing that occurs when the mouse moves is to update any active cursors being drawn. This processing does not occur if 0 is returned as the result. Instead, any active cursors are removed from the display. The default implementation returns 1.

on_leave_window(event)

This method is called when the user moves the mouse out of the **PlotWindow** object. Refer to the **on_left_down** method for more information about **event** and the result returned.

The normal event processing that occurs when the pointer leaves a **PlotWindow** is to hide any cursors that are active. This processing is not performed if 0 is returned as the result. The default implementation returns 1.

on_right_click(event)

This method is called when the user clicks the right mouse button over the **PlotWindow** object. Refer to the **on_left_down** method for more information about **event** and the result returned.

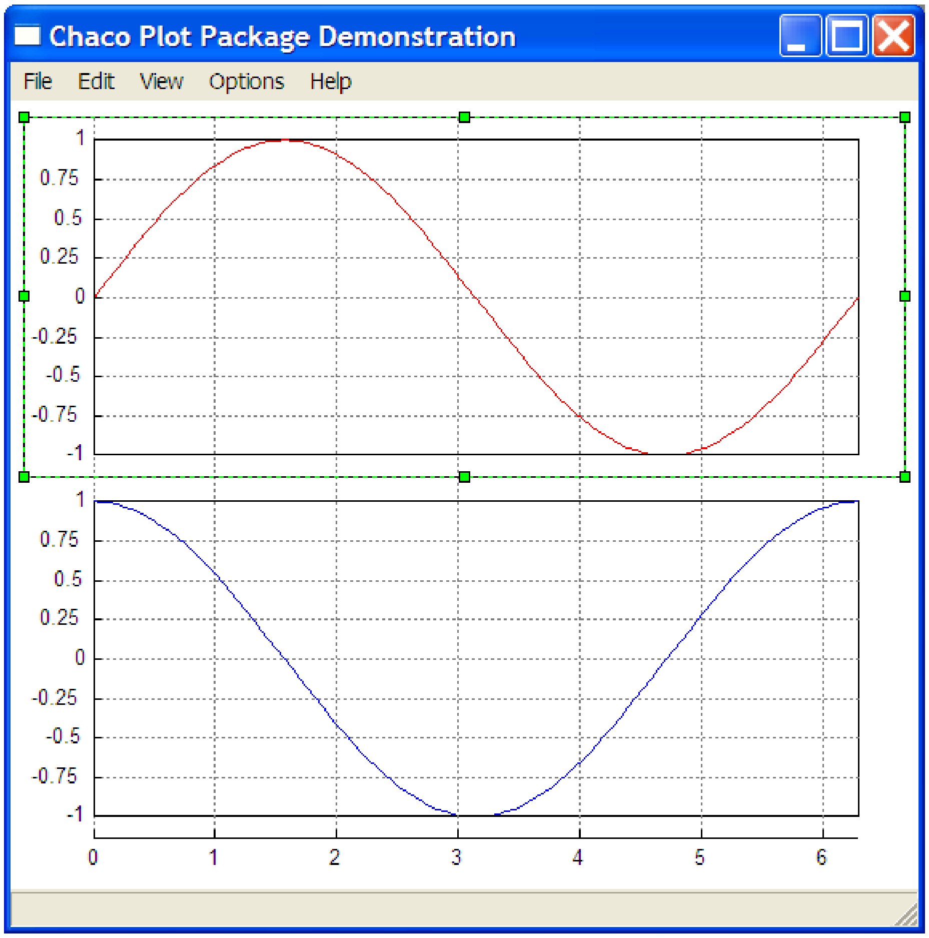
The normal event processing that occurs when the right mouse button is clicked is to display the appropriate contextual pop-up menu or trait editor for the Chaco object (or objects) under the pointer. This processing does not occur if 0 is returned as the result. The default implementation returns 1.

A **PlotWindow** is also a Chaco **DefaultContainer** object and acts as the container for the Chaco object assigned to the **PlotWindow's canvas** trait. Therefore a **PlotWindow** has all of the traits defined for a **DefaultContainer**. Refer to the **DefaultContainer** section for more information on what those traits are.

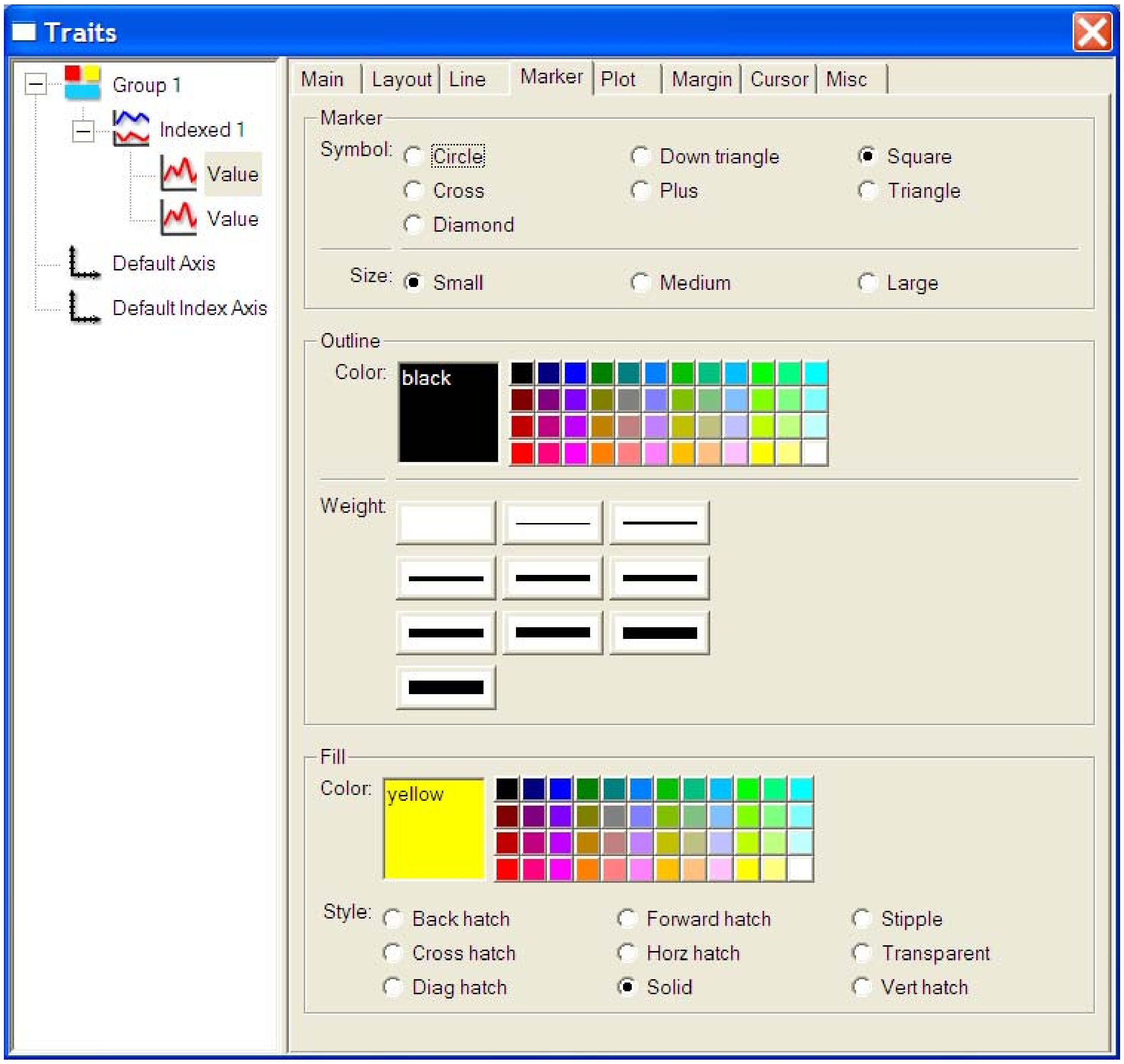
Trait Editors

Another benefit of using the **traits** package is the ability to provide high-quality, multi-GUI (wxPython and Tkinter currently) interactive trait editors.

One of the optional features of the Chaco package is the ability to allow the user to visually select and edit the properties of any Chaco object interactively. The following figure shows one of the previous examples running in the Chaco demo program with one of the **PlotValue** objects selected:



The next figure shows one page from the trait editor displayed by double-clicking on the selected object:



Marker

Symbol: Circle Down triangle [✓ Square] Cross Plus Triangle Diamond

Size: [✓ Small] Medium Large

Outline

Color: black

Weight:

Fill

Color: yellow

Style: Back hatch Forward hatch Stipple Cross hatch Horz hatch Transparent Diag hatch [✓ Solid] Vert hatch

Changing any of the trait values in the editor is immediately reflected in the displayed **PlotValue's** on-screen representation.

Note also that the trait editor displays the complete hierarchy of all Chaco objects contained in the window. New objects can be selected either by clicking on their label in the hierarchy view or by clicking on the object itself in the plot window. In either case, the editor notebook on the right side of the dialog immediately reconfigures itself to display the newly selected object's traits.

Plot Renderers

Up until now, each of the plots displayed has been a simple line plot of the associated data. While this is probably the most common type of plot for many scientific and engineering applications, a plot package that supported only one type of plot would not be very useful. To that end, Chaco supports an open-ended plotting architecture based on the notion of *plot renderers*.

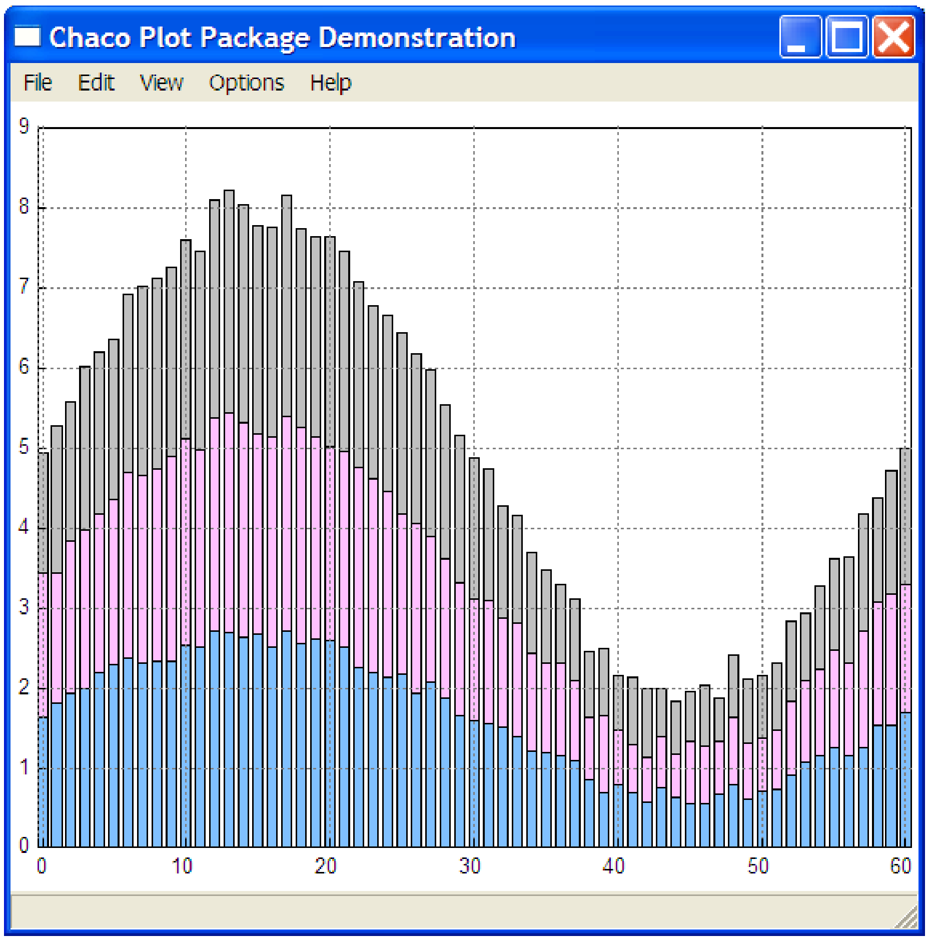
Each **PlotValue** has a **plot_type** trait that defines a set of one or more plot renderers that will be used to render (i.e. draw) the associated **PlotValue’s** data. The default value of this trait is ‘line’, which specifies the line rendering style we have seen so far in all of the examples.

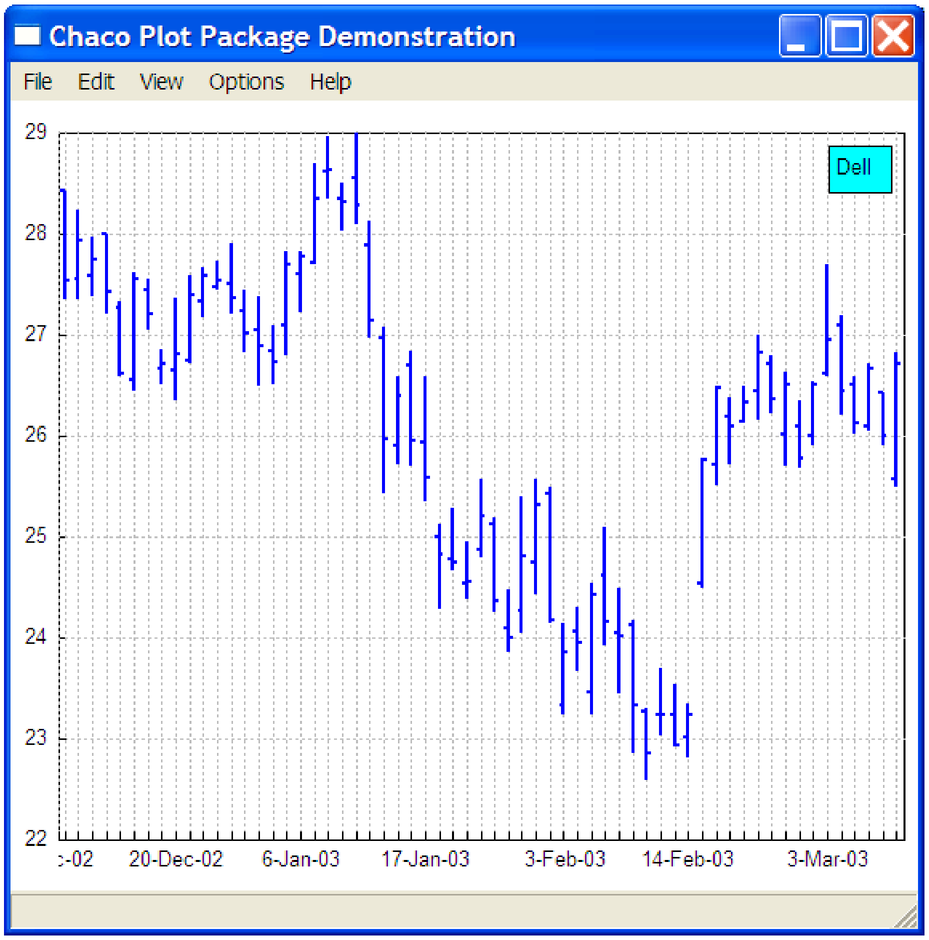
However, the following are some examples of other values that could be set:

```
value = PlotValue( … )
value.plot type = ‘scatter’         # Scatter plot
value.plot type = ‘scatter,line’    # Both scatter and line
value.plot_type = [ ‘scatter’, ‘line’ ] # Same thing
```

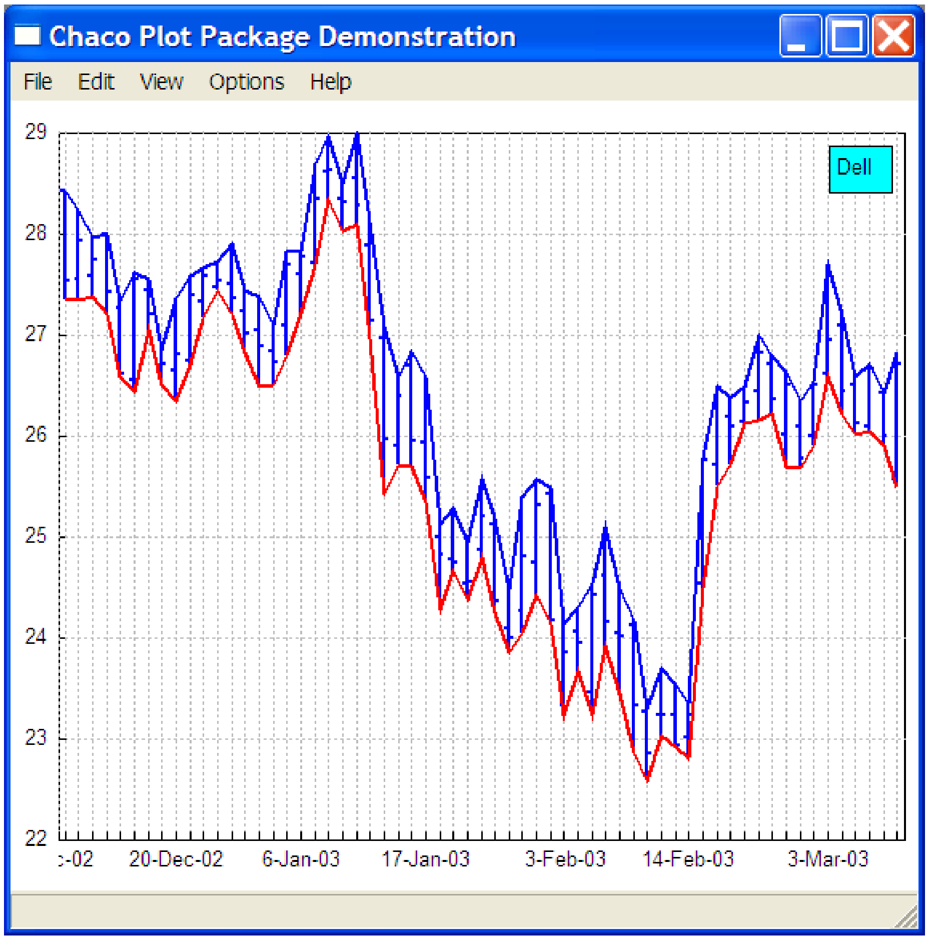
Note that more than one type of plot renderer can be specified. Also, new types of plot renderers can be added to Chaco simply by subclassing the **Renderer** base class. More information on how to do this will be provided in a later section.

Another interesting feature of the plot renderer architecture is that a given **Renderer** subclass can plot several related sets of data at the same time. Examples of this type of plot are *stacked bar charts* and stock (i.e. equity) *HLOC* (high/low/open/close) diagrams, illustrated in the following figures:





As mentioned previously, more than one plot renderer can be associated with a **PlotValue**, as illustrated in the following figure, which combines the *HLOC* renderer shown above with the standard *line* renderer:



Creating a Renderer Subclass

To create a new Chaco renderer, you must subclass **Renderer** or one of its subclasses. Also, if you want users to be able to specify your renderer by name, there are some additional rules that must be followed in naming both your subclass and your module:

- In addition to being a subclass of **Renderer**, your class must also have a name of the form: ***xxx*Renderer**, where ***xxx*** is the name a user will use to identify your renderer (e.g. **Polar**). Note that spelling is important, but case is not (i.e. calling your class **PolarRenderer** works the same as calling it **polarrenderer**).

- You must call your module ***xxx*.py**, where ***xxx*** is the same name you used to name your class, with the additional restriction that it must be all lower case (e.g. **polar.py**).

- Finally, the module must be stored in the **chaco.renderers** package directory. This is necessary because Chaco attempts to load the module as if it were part of the Chaco package.

Note that you do not need to follow the above naming guidelines if users of your renderer specify either the renderer’s class or an instance of the renderer. For example:

```
# User specifies name, must follow naming guidelines:
my plot.type = ‘polar’

# User provides class, no need to follow guidelines:
my plot.type = MyPolarRendererClass

# User provides instance, no need to follow guidelines:
my_plot.type = MyPolarRendererClass()
```

In many cases, the best place to start when writing a new renderer is with an existing one, either to see how one is written, or to use it as a basis for subclassing.

The base **Renderer** base class defines all of the methods that a subclass must implement. In many cases, a default implementation is provided that works for many rendering types, so it is not necessary to override every method to define a new renderer.

The following are the important **Renderer** methods that a subclass must implement:

_max_values

Returns the maximum number of values the renderer can process. A value of 0 (the default) means the renderer can handle an unlimited number of values.

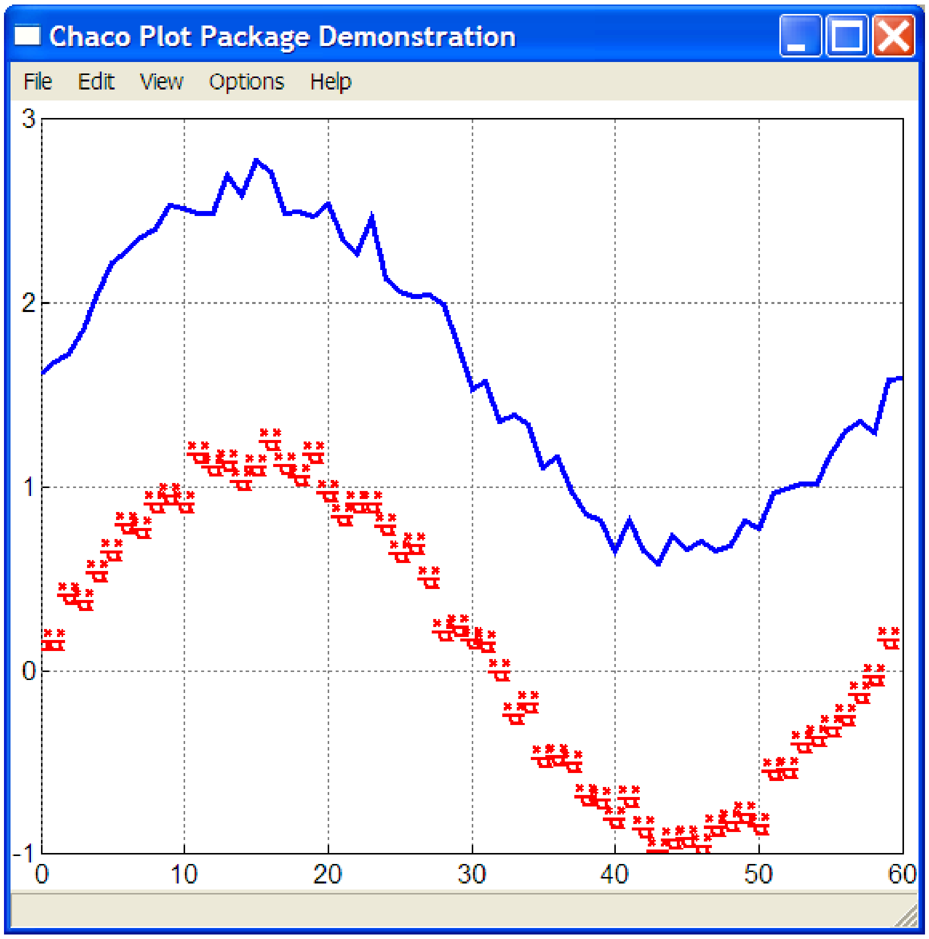
Some renderers are designed to handle a specific number of values. For example, the **HLOC** renderer can handle a maximum of four values (i.e. a *high*, *low*, *open* and *close* value). Other renderers, such as the default **line** renderer, can handle an unlimited number of values, since each value is rendered independently of the others. By specifying a value > 0, Chaco will ensure that your renderer is never passed more values than you specify as the maximum.

_is_valid (values)

Determines if the renderer can correctly plot a set of values. The **values** argument is a list of **PlotValue** objects to be plotted. The method should return a non-zero value if the values can be correctly plotted, and 0 otherwise. The default implementation returns 1 to indicate that the **values** can be plotted.

Some renderers may have certain preconditions that need to be satisfied in order for a set of values to be plotted. This method gives them the opportunity to ensure that those preconditions are met. For example, the stacked bar chart renderer can plot an unlimited number of values, but requires that all elements be >= 0. If it finds any negative values in the elements it receives, it returns 0 to indicate that it cannot correctly plot the data.

Chaco visually indicates an error condition to the user by replacing the requested renderer with the **invalid** renderer, which plots error markers to highlight the values in error. This is illustrated in the following figure, which shows the result when attempting to plot a **PlotValue** with negative data points using the stacked bar chart renderer:



_get_bounds_value (values)

Returns a tuple containing the range (i.e. (min, max)) for the value part of the list of **PlotValues** specified by **values**. This method does not normally need to be overridden, since the default implementation determines the range by examining the list of **PlotValues**. However, it can be overridden in special cases where the range depends on more than just the contents of the **PlotValue** objects being plotted

_get_bounds_index (values)

Returns a tuple containing the range (i.e. (min, max)) for the index part of the list of **PlotValues** specified by **values**. This method does not normally need to be overridden, since the default implementation determines the range by examining the list of **PlotValues**. However, it can be overridden in special cases where the range depends on more than just the contents of the **PlotValue** objects being plotted

_layout (values, x, y, dx, dy, sx, sy)

Layout a list of **PlotValues** specified by **values** into the plotting region specified by **x**, **y**, **dx**, **dy**, using the scaling factor specified by **sx** and **sy**.

Chaco divides plotting into several phases, including layout and drawing. During the layout phase, a renderer is given the opportunity to calculate and cache information that it can use during the drawing phase to speed up drawing of the specified values. This typically includes such things as transforming and caching data space coordinates into plot space coordinates. The renderer can cache information in each **PlotValue** object using attribute names beginning with ***_xxx_***, where ***xxx*** is the renderer name described previously (e.g. **polar**). Following this convention is important because multiple renderers can be applied to the same **PlotValue**, and each may need to cache different types of information.

The **x**, **y** values specify the lower left hand corner of the plotting area, while **dx** and **dy** specify its width and height respectively. The **sx** and **sy** values specify scaling factors to be applied when mapping from data space to plot space. A scaling factor of 1.0 means that all of the data should exactly fit into the plotting region specified. A scaling factor of 2.0 means that the data should be mapped into a region twice the size of the region actually specified, and so on. Note that a scaling factor will never be < 1.0.

_draw_canvas (values, ox, oy)

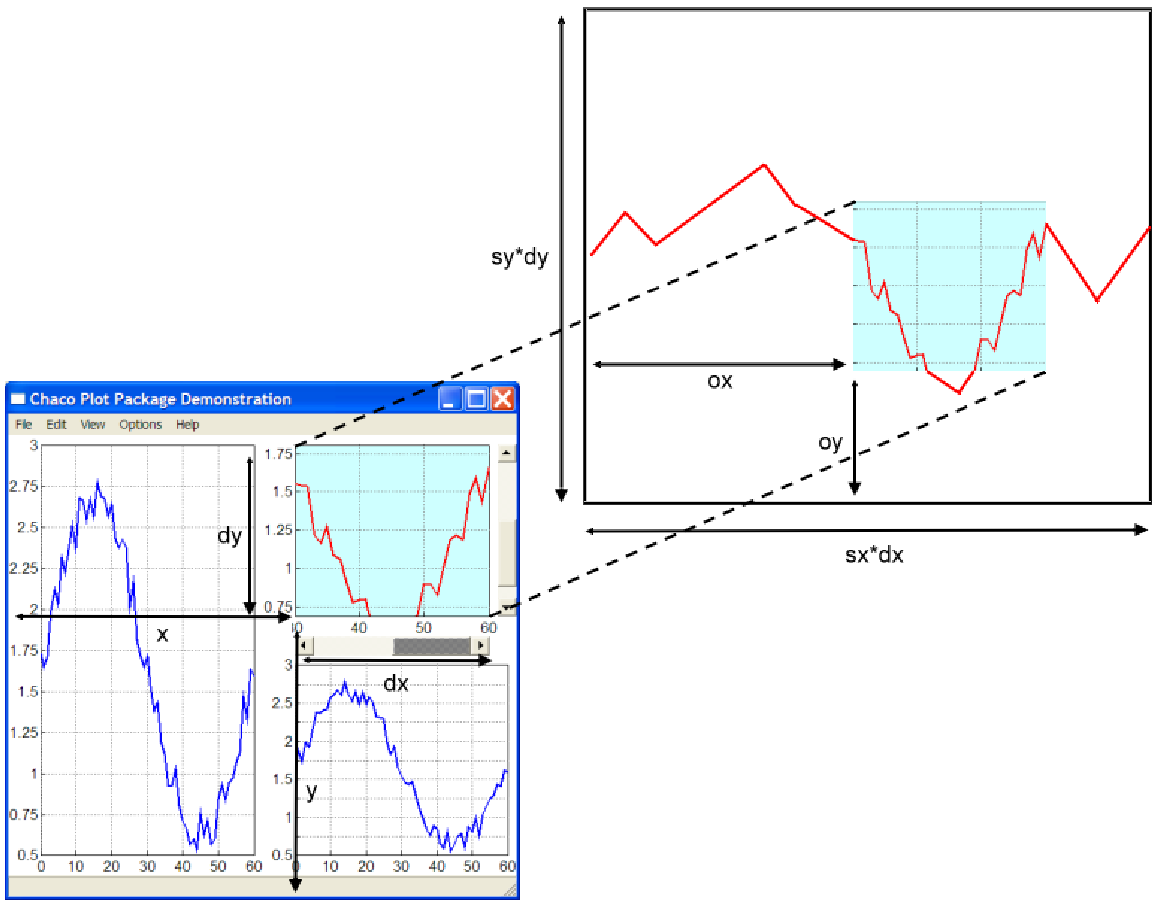
Draw the *background* portion of the list of **PlotValues** specified by **values**, using the offsets specified by **ox** and **oy**.

Chaco divides drawing into two phases: *background* and *foreground*. During the background phase, the renderer should draw whatever portion of the plot should appear *underneath* all other portions of the plot, including those drawn by other renderers on the same canvas.

An example of this is provided by the **fill** renderer, which fills the region between two sets of values. In its **_draw_canvas** method (the background phase) it draws the fill regions between pairs of values. In its **_draw** method it draws the lines defining the *edges* of the fill regions. Doing it this way allows other types of plots, such as simple line plots, to overlay the fill plot without being obscured by the filled areas.

It is a design decision on the part of the renderer developer as to what parts of the plot should be drawn in the **_draw_canvas** method, and what parts should be drawn in the **_draw** method.

The **ox** and **oy** values specify the point in the *scaled* plot region that should be mapped to the origin of the *physical* plot region. These values interact with the **x**, **y**, **dx**, **dy**, **sx** and **sy** values specified in the **_layout** method called before drawing begins, as shown in the following diagram:



_draw (values, ox, oy)

Draw the *foreground* portion of the list of **PlotValues** specified by **values**, using the offsets specified by **ox** and **oy**. Refer to the description of the **_draw_canvas** method above for a description of the difference between *foreground* and *background*, and the meaning of the various method arguments.

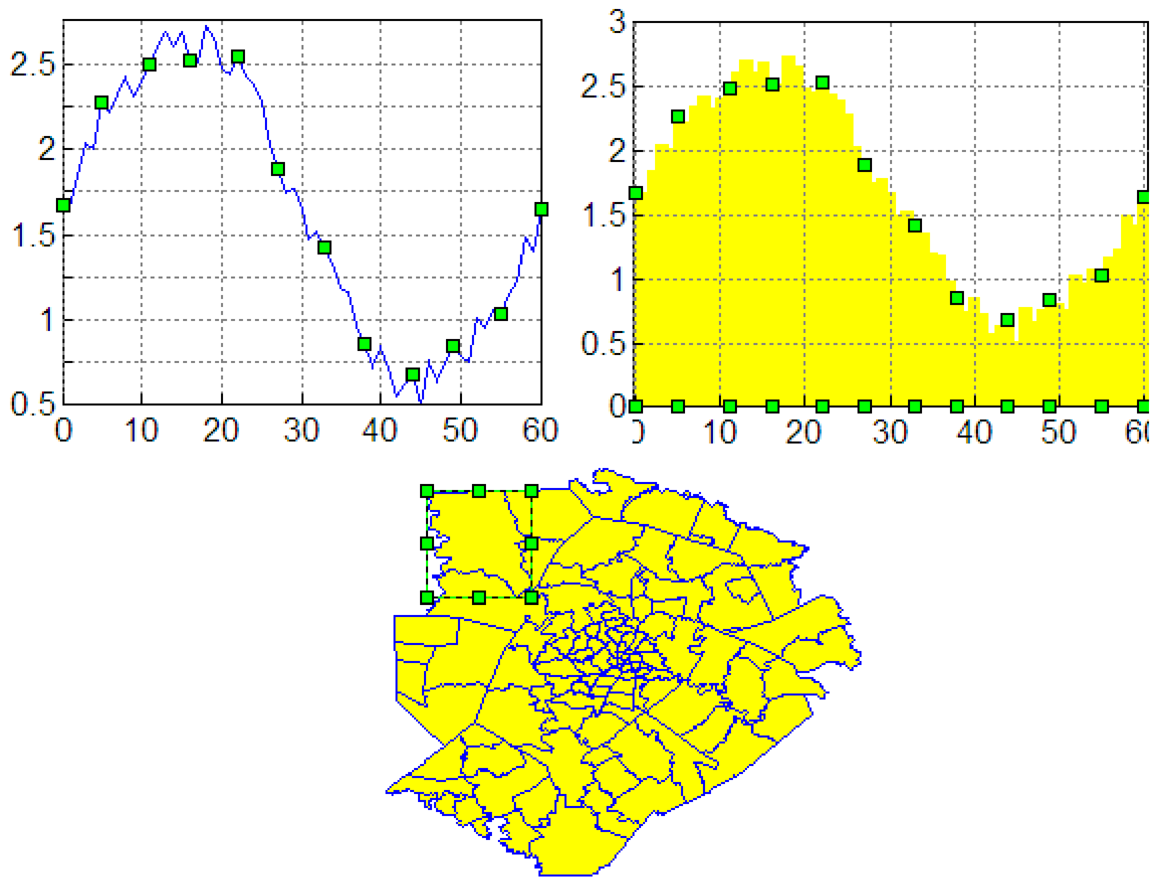
_draw_plot (value, ox, oy)

Draw the *foreground* portion of the single **PlotValue** object specified by **value**, using the offsets specified by **ox** and **oy**.

If your renderer does not need to coordinate the drawing of several **PlotValue** objects at the same time, you may find it easier to override the **_draw_plot** method rather than the **_draw** method. The default **_draw** method simply calls **_draw_plot** once for each **PlotValue** it receives.

_draw_selection (values, ox, oy)

Draw the selection markers for any selected **PlotValue** objects in the list specified by **values**. A *selection marker* is the special border drawn around objects to indicate that they have been selected. The following figure shows several different types of selection markers drawn by various Chaco renderers:



Unless you have special requirements for drawing a selection marker, you should not need to override this method, since the base class method will automatically draw the appropriate selection marker based upon characteristics of the selected **PlotValue** object's index and data values:

- If the index values of the **PlotValue** object are monotonically increasing or decreasing, it will draw one of the top two selection markers shown above. It draws the type shown in the top-left part of the figure if the renderer's **select_zero** attribute is zero (the default for the **Renderer** base class). Otherwise, if **select_zero** is not zero, it draws the selection marker shown in the top-right part of the figure.

- If the index values of the **PlotValue** object are not ordered, then it draws the box selection marker shown in the bottom part of the figure.

If you do override the method, you should only draw selection markers for **PlotValue** objects in **values** whose **selected_** attribute is non-zero. The **ox** and **oy** values have the same meaning as they do for the **_draw_canvas** and **_draw** methods described previously.

_hittest (values, x, y)

Determine if the visual representation of any of the **PlotValue** objects in the list specified by **values** intersects (or lies within a small delta of) a specified (**x**,**y**) point.

The **x** and **y** values are in plot space coordinates. Thus, in order to correctly perform the hit test it is usually necessary for the renderer to cache the current plot area origin (**x**,**y**) and size (**dx**,**dy**) as well as the offset (**ox**,**oy**) specified in the most recent layout and drawing method calls.

The method should return a list of all **PlotValue** objects in **values** whose visual representation intersects or lies within a small delta of the specified point.

Note that there is no default implementation for this method in the base **Renderer** class.

_map_xy (value, x, y)

Map a specified (**x**,**y**) point in plot space to the corresponding point (x',y',v') in the data space of the **PlotValue** object specified by **value** (v' is the **PlotValue** intercept of a line orthogonal to the index axis that passes through the point (x',y')).

As with the **_hittest** method, in order to correctly implement this method it is usually necessary for the renderer to cache the current plot area origin (**x**,**y**) and size (**dx**,**dy**) as well as the offset (**ox**,**oy**) specified in the most recent layout and drawing method calls.

The result returned by the method should be the computed (x',y',v') tuple.

PlotData and PlotSource Objects

As mentioned previously, a **PlotValue** object can be logically viewed as simply an array of values to be plotted. While this is a very convenient model in many cases, especially when working interactively in a Python command shell, it is not always the best model when working with more structured data, such as the stock *high*, *low*, *open*, and *close* data we used in the previous example.

Accordingly, Chaco also provides a means of dealing with more structured forms of data through the use of **PlotData** and **PlotSource** objects. A **PlotData** object is simply a wrapper around one or more sets of related data. The data contained in the **PlotData** object can be connected to a plottable Chaco object by means of a **PlotSource** object, which acts as a conduit between a **PlotValue** and a particular set of data contained within a **PlotData** object.

A good example of a **PlotData** object is the **StockQuote** class we used in the previous example. A **StockQuote** object contains a particular stock's *open*, *close*, *high*, *low* and *volume* data over a specified range of dates. The data is obtained dynamically over the Web whenever the **StockQuote** object's **symbol** trait is modified:

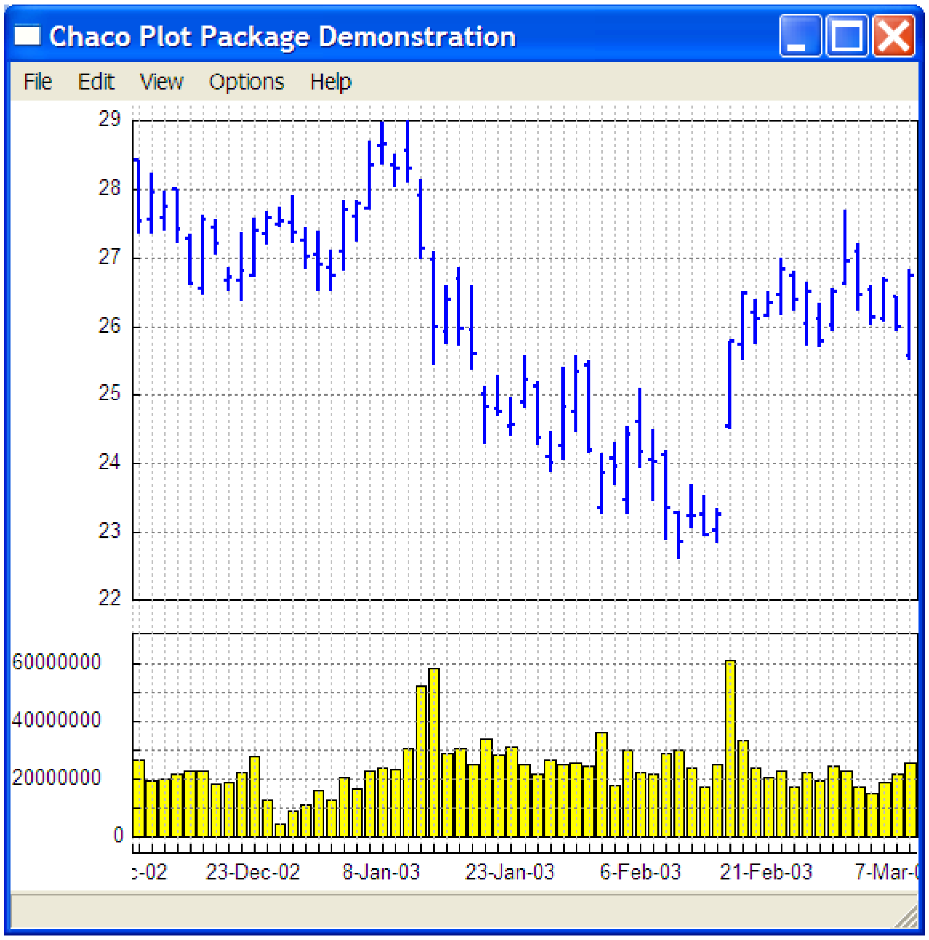
```
sq = StockQuote()
sq.symbol = 'intc'  # Fetches new values for symbol 'intc'
```

A **StockQuote** object also exposes six sources of data: **high**, **low**, **open**, **close**, **volume** and **date** (the *index* values for the other five data sources), any of which can be associated with one or more **PlotValue** objects. **PlotData** objects can also be *smart* about how they add themselves to a plottable object. For example, the preceding stock quote examples showing the *HLOC* plots were constructed using a statement similar to the following:

```
canvas.add( StockQuote( 'dell' ) )
```

In this case, the **StockQuote** object automatically connected its *high*, *low*, *open*, *close* data sources to the **PlotCanvas** object and set the **plot_type** trait of the canvas to 'HLOC' to select the correct default plot renderer.

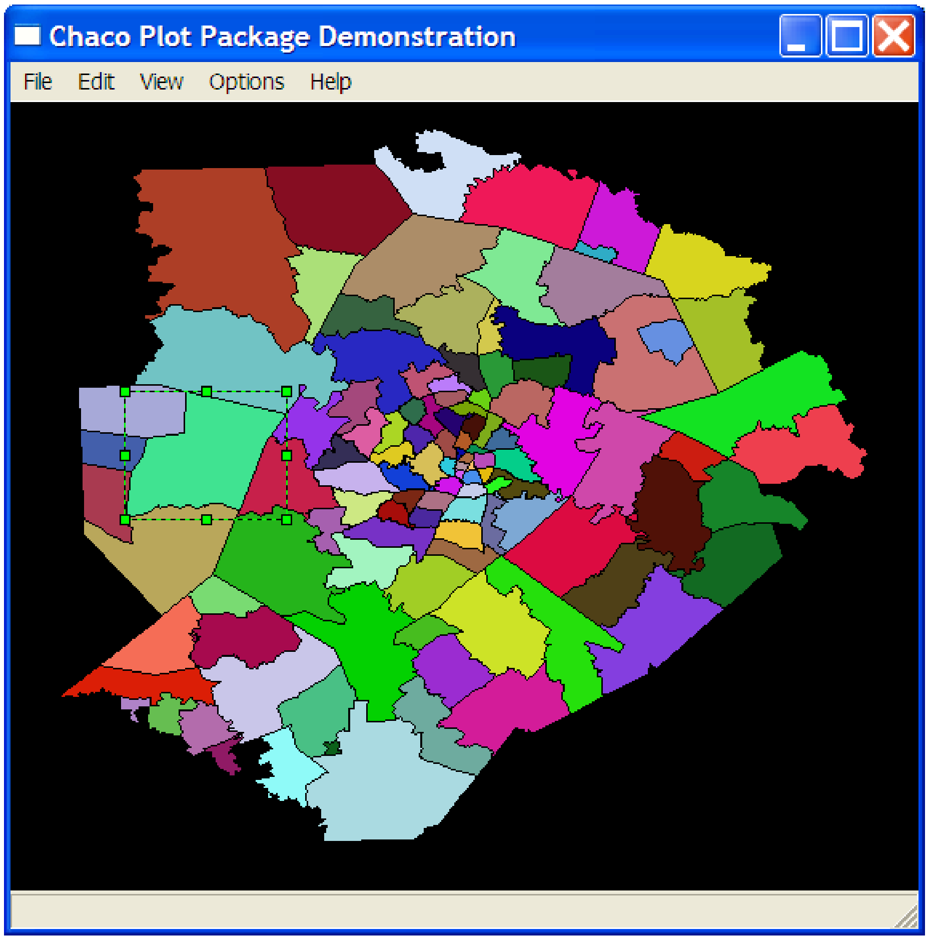
We can also explicitly connect a **PlotData** object source to a particular plottable object directly, as illustrated in the following figure:



where we have explicitly connected the *volume* source of the **StockQuote** object to the bottom **PlotCanvas** using a statement like:

```
sq.add( canvas, ‘volume’ )
```

Another example of the usefulness of the **PlotData** capability of Chaco is shown in the following figure, which was constructed using an instance of the **ArcView PlotData** subclass:



An **ArcView** object loads GIS data in ***ArcView*** format from a data file and provides each of the polygons it contains as a separate data source. When we execute a statement like the following:

```
canvas.add( arcview_object )
```

the **ArcView** object (i.e. **arcview_object**) adds each of its polygons to the canvas as a separate **PlotValue** object and sets the **PlotCanvas** object’s **plot_type** trait to ‘polygon’, which is the appropriate plot renderer for this type of data.

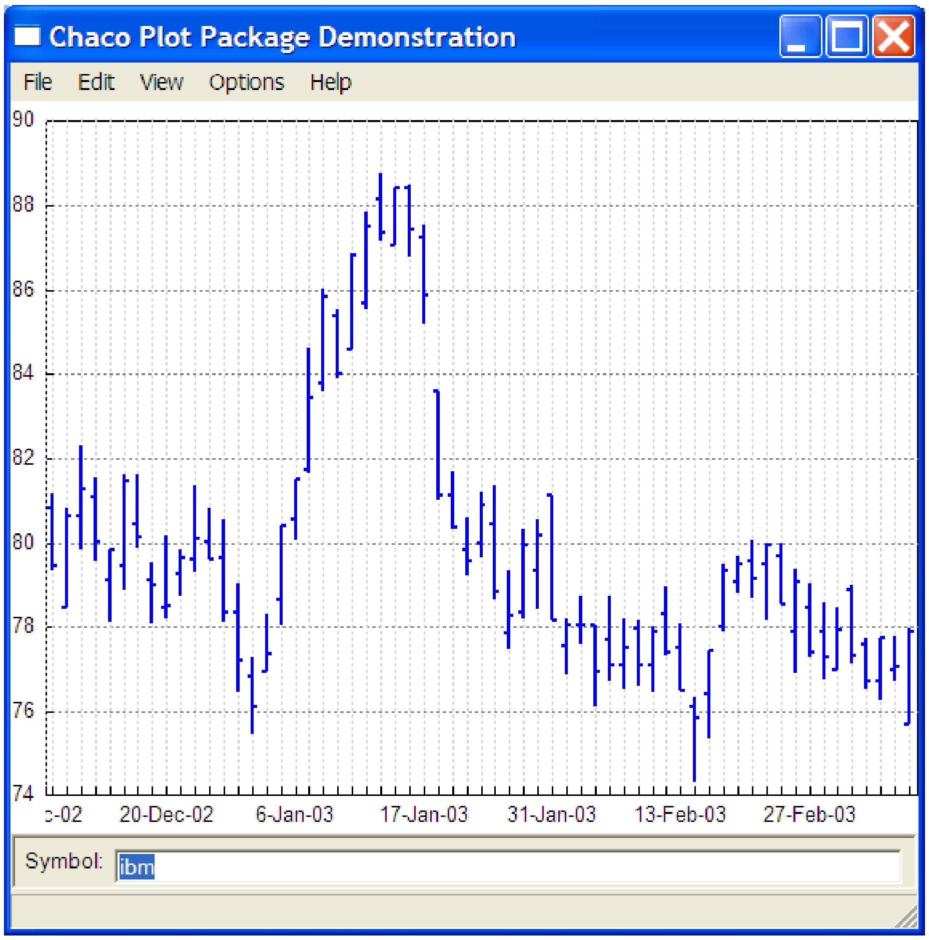
The fill colors for this example were set randomly as follows:

```
for value in canvas[0]:
    value.fill_color = (random(), random(), random(), 1.0)
```

Note also that one of the polygons (i.e. **PlotValues**) on the canvas has been selected in the figure. Double-clicking the polygon would display its trait editor, allowing the user to interactively change any of the object's traits, such as selecting a new fill color.

PlotPanel Objects

Another feature of Chaco is its ability to imbed panels containing various useful widgets within a **PlotWindow**. An example of this is shown in the following figure:



This view was constructed using the following statements:

```
sq = StockQuote( symbol = 'ibm' )
group.add( PlotCanvas( sq ), PlotPanel( sq ) )
```

When added to a **PlotCanvas**, the **StockQuote** object binds its *open*, *close*, *high* and *low* data sources to **PlotValue** objects added to the canvas. When added to a **PlotPanel**, a trait editor for the **StockQuote** object’s traits (in this case, its stock symbol name) is automatically constructed and imbedded within the panel.

Furthermore, the resulting **Symbol** field is live. Typing a new stock symbol into the text entry field results in a new set of stock values being displayed. It should be further noted that there is no explicit code in the implementation of the **StockQuote** class to do this.

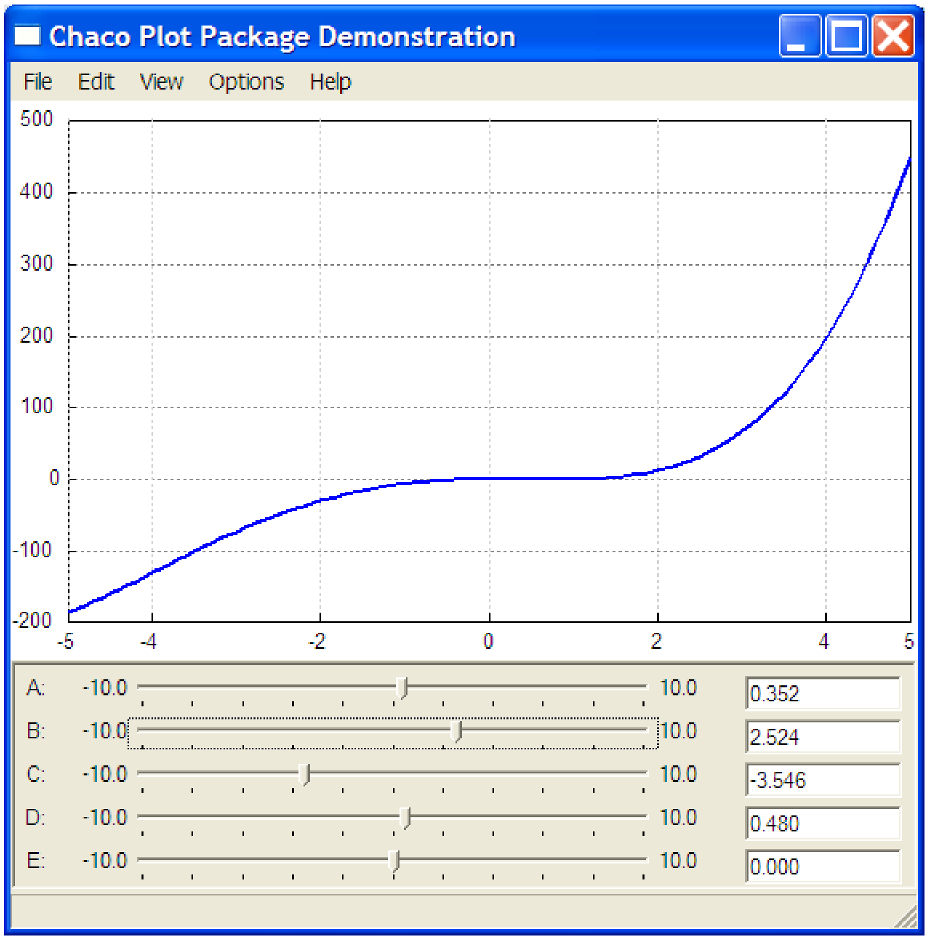
This latter point is illustrated in the following example, which shows the code for a simple **Polynomial** class for displaying fifth degree polynomials:

```
class Polynomial ( PlotData ):

   coefficient = Trait( 0.0, TraitRange( -10.0, 10.0 ) )

     traits   = {
      'a': coefficient,
      'b': coefficient,
      'c': coefficient,
      'd': coefficient,
      'e': coefficient
   }

   def data ( self, source name ):
       x = arange( -5.0, 5.01, 0.1 )
       y = ((self.a * (x ** 4)) +
            (self.b * (x ** 3)) +
            (self.c * (x ** 2)) +
            (self.d * x) + self.e)
       return transpose( reshape( concatenate( ( x, y ) ),
                         ( 2, len( x ) ) ) )
```

With the exception of a few **import** statements, this is the entire definition of the **Polynomial** class. We can now construct a view containing an instance of this class using code very similar to the preceding example:

```
poly = Polynomial()
group.add( poly, PlotPanel( poly ) )
```

This results in a window looking something like the following:



This figure shows the results achieved after adjusting several of the panel sliders, each of which corresponds to one of the polynomial coefficients. Again, it is interesting to note that there is no code in the **Polynomial** class that deals explicitly with changes to the polynomial coefficients. All of the "magic" occurs under the covers, courtesy of the **traits** package.

Creating a PlotData Subclass

You can create your own type of *smart* data objects by subclassing **PlotData** and overriding some or all of the following methods:

data (source_name)
Return the array or list of data corresponding to a specified **source_name**. This method *must* be overridden by a subclass. The class is free to define the set of **source_name** strings that it understands.

source (value, source_name)
Creates and returns a new **PlotSource** subclass instance used to connect the specified **source_name** and **PlotValue value** object. You only need to override this method if you define you own **PlotSource** subclass.

add_to (plottable, source_name)
Adds one or more source connections, specified by **source_name**, to a **plottable** object. This method can be overridden by a subclass.

The **data** method returns an array of data corresponding to the specified **source_name** argument. For example, in the **StockQuote** sample, the possible values for **source_name** are: **high**, **low**, **open**, **close**, **volume** and **date**. If your class only has one source of data, the **source_name** argument can be ignored.

The result returned by **data** can either be a one-dimensional array of values, or a two-dimensional array of **(index, value)** pairs. Although the *values* should always be numbers, the *indices* can either be all numbers or all strings. In all cases, the result returned by **data** represents a set of values to be plotted.

The **source** method only needs to be overridden if your **PlotData** subclass has an associated **PlotSource** subclass. In this case, the **source** method should return an instance of your **PlotSource** subclass.

The **add_to** method only needs to be overridden if you need special processing to occur when your object is added to a Chaco container. For example, the **StockQuote** sample handles the case where the **source_name** is **None** and **plottable** is a **PlotCanvas** object by creating a **PlotValue** object for each of its *high*, *low*, *open* and *close* data sources and adding them to the specified canvas. It also sets the canvas's **plot_type** trait value to **HLOC**.

Creating a PlotSource Subclass

A **PlotSource** instance connects a **PlotValue** instance to a particular **PlotData** object data source. Among other things, it helps synchronize changes in the data source to the associated **PlotValue** object. In general, there is no reason to modify the behavior of the base **PlotSource** class.

The only reason to create a new **PlotSource** subclass is to specify the set of legal source names that can be used with its associated **PlotData** object. For example, this is the **PlotSource** subclass created for use with the **StockQuote** sample:

```
class StockQuoteData ( PlotSource ):

   __traits__ = {
      'source': [ 'close', 'open', 'high', 'low',
                  'volume', 'date' ]
   }
```

The only thing contained in the class definition is a new definition for the **source** trait which enumerates its associated **StockQuote** class's data source names. In general, each string specified in the **source** trait definition corresponds to a valid **source_name** value for its associated **PlotData** subclass's **data** method.

Kiva and the Render Method

Chaco implements all of its plotting capabilities using **kiva**, a DisplayPDF based drawing package that supports multiple output types. As a result of this, any composition of Chaco objects can be rendered to any of the following output formats:

- wxPython or Tkinter based display screen windows
- PDF
- PostScript
- Raster images such as JPEG, PNG or GIF

This flexibility in output formats makes Chaco potentially useful in a wide variety of applications, including:

- Ad-hoc plots performed from a Python interpreter shell.
- Interactive wxPython or Tkinter based applications requiring plotting capabilities.
- On-demand production of plot related images by a web server CGI script.
- Compositing of plots with other forms of text and graphics within a PDF or PostScript based report generator.

An application written using either the wxPython or Tkinter based Chaco support normally does not need to be aware of **kiva**, since the **PlotWindow** class manages the **kiva** interface for you.

However, if you're writing a noninteractive application that will render Chaco output to formats such as SVG or PDF, you'll need to create a kiva graphics context and use it in conjunction with the Chaco **render** method.

The **render** method is defined on all Chaco container objects. It renders the container object and all of its contents into the specified kiva graphics context. Normally the **render** method is only used with the outermost container object, since the container object will automatically render all of its contents. However, you can also apply **render** to any nested container object to render just that object and its contents. This is illustrated in the Chaco demo program, for example, which provides an option to render each individual plot to a separate PDF page.

The form of the **render** method is:

render(gc = None, x = 0, y = 0, dx = 640, dy = 480)
Renders the receiving container object and all of its contents to the kiva graphics context specified by **gc**.

The **gc** parameter specifies the kiva graphics context to render to. If omitted or **None**, it defaults to the current graphics context associated with the container object. By default there is no context associated with a container object, so you should normally specify one in your call to **render**.

The **x**, **y**, **dx** and **dy** parameters specify the position and size of the area within the graphics context that the container object and its contents are rendered to.

For example, the following code illustrates rendering a very simple plot to a PDF file using the kiva **pdfcore2d.py** module:

```
import kiva.pdfcore2d
gc     = kiva.pdfcore2d.Canvas( 'simple.pdf' )
dx, dy = gc.size()
value  = PlotValue( [ 1, 3, 2 ] )
value.render( gc, 20, 20, dx - 55, dy + 10 )
gc.end page()
gc.save()
```

For more information on creating and using other types of kiva graphics contexts, refer to the kiva documentation or the source module for the kiva graphics driver you are interested in using.

Where To Get The Code

The **Chaco** package is available as part of the **SciPy** (***Sci***entific ***Py***thon) open source project at **www.scipy.org** and is covered by a BSD-style license. All of the code described in this document is contained in the **chaco** project in the www.scipy.org CVS repository.

The CVS repository can be accessed from a command shell using the commands:

```
> cvs -d:pserver:anonymous@scipy.org:/home/cvsroot login
> cvs -d:pserver:anonymous@scipy.org:/home/cvsroot co
chaco_all
```

Note the use of **chaco_all** as the CVS module name. Checking out this module ensures that all of the Chaco code and its supporting packages, such as **kiva** and **traits**, are copied from the CVS repository.

The Chaco source code can also be browsed on-line using the **ViewCVS** facility. There is also a Chaco page at **http://www.scipy.org/site_content/chaco** that contains recent Chaco source tarballs and a Windows installer program.