Plotting graphs in SVG format

Jake Voytko
Paul A. Bristow

Copyright © 2007 to 2009 Jake Voytko and Paul A. Bristow

Distributed under the Boost Software License, Version 1.0. (See accompanying file LICENSE_1_0.txt or copy at http://www.boost.org/LICENSE_1_0.txt)

Table of Contents

Preface ... 2
How To Use This Documentation ... 3
Colors ... 4
Fonts ... 6
1D Tutorials ... 11
- 1-D Vector Example ... 11
- 1-D STL Containers Examples ... 13
- Tutorial: 1D Autoscale with Multiple Containers ... 15
- Tutorial: 1D More Layout Examples ... 18
- Tutorial: 1D Gridlines & Axes - more examples ... 20
- 1-D Axis Scaling ... 25
- 1-D Auto scaling Examples ... 29
- 1-D Autoscaling Various Containers Examples ... 38
- 1-D Data Values Examples ... 41
- Real-life Heat flow data ... 46
- Demonstration of using 1D data that includes information about its Uncertainty ... 52

2D Tutorial ... 55
- Simple Code Example ... 55
- Tutorial: Fuller Layout Example ... 57
- Tutorial: 2D Special Features ... 59
- 2-D Data Values Examples ... 65
- 2-D Autoscaling Examples ... 70
- Demonstration of using 1D data that includes information about its uncertainty ... 74
- Demonstration of adding lines and curves, typically a least squares fit ... 76
- Histograms of 2D data ... 78

Tutorial: Boxplot ... 80
- Simple Example ... 80
- Definitions of the Quartiles ... 82

SVG tutorial ... 86
Showing data values at Numerical Limits ... 86
Reference ... 87
- Header <boost/svg_plot/detail/auto_axes.hpp> ... 87
- Header <boost/svg_plot/detail/axis_plot_frame.hpp> ... 103
- Header <boost/svg_plot/detail/FP_compare.hpp> ... 105
- Header <boost/svg_plot/detail/functors.hpp> ... 107
- Header <boost/svg_plot/detail/numeric_limits_handling.hpp> ... 108
- Header <boost/svg_plot/detail/pair.hpp> ... 108
- Header <boost/svg_plot/detail/svg_boxplot_detail.hpp> ... 108
- Header <boost/svg_plot/detail/svg_style_detail.hpp> ... 109
- Header <boost/svg_plot/detail/svg_tag.hpp> ... 109
- Header <boost/svg_plot/quantile.hpp> ... 180
- Header <boost/svg_plot/show_1d_settings.hpp> ... 182
- Header <boost/svg_plot/show_2d_settings.hpp> ... 185

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Header <boost/svg_plot/stylesheet.hpp> 186
Header <boost/svg_plot/svg.hpp> 189
Header <boost/svg_plot/svg_1d_plot.hpp> 197
Header <boost/svg_plot/svg_2d_plot.hpp> 230
Header <boost/svg_plot/svg_boxplot.hpp> 278
Header <boost/svg_plot/svg_color.hpp> 327
Header <boost/svg_plot/svg_fwd.hpp> 337
Header <boost/svg_plot/svg_style.hpp> 338
Header <boost/svg_plot/uncertain.hpp> 370
Implementation & Rationale 384
To Do List 388
Acknowledgements 389
Class Index 390
Function Index 444
Index 498

Preface

Preface

Humans have a fantastic capacity for visual understanding, and merely looking at data organized in one, two, or three dimensions allows us to see relations not otherwise visible in a list of numbers. Computers, however, deal with information numerically, and C++ and the Standard Template Library (STL) do not currently offer an easy way to bridge the gap.

This library aims to help the user to easily plot data that is stored in STL containers.

This project is focused on using STL containers in order to display data on a one-dimensional, two-dimensional plot and boxplots. The plot is written to a Scalable Vector Graphic image (that can also be converted to other graph types, for example, PNG http://en.wikipedia.org/wiki/Portable_Network_Graphics).

Scalable Vector Graphics (SVG) is an XML specification and file format for describing two-dimensional vector graphics.

SVG files (.svg) can be viewed with an Internet Browser:

- Mozilla Firefox (version 3 and later)

You may wish to associate Firefox with SVG files (or perhaps Inkview - see below).

- Adobe Illustrator.
- Opera has svg support but the quality of rendering is sometimes not as good as other browsers.
- Microsoft Internet Explorer, provided a suitable Adobe SVG Viewer plug-in for SVG files is installed. Shamefully, even IE8 Beta does not yet offer SVG rendering. (Adobe have stopped offering this for download (as they warned years ago) but the download software can still be obtained elsewhere, for example from software.informer. Information about limitations of IE with Vista and the Adobe add-in are given on the Adobe End of Line FAQ).
- Inkscape, a fine Open Source SVG editor Inkscape and a viewer Inkview with excellent rendering, full scaling and other editing features.

Inkview also allows window resizing, showing the scalability of SVG files without any loss of quality, unlike bit image files that eventually reveal pixels as you expand the view.

You may find it especially convenient to use the viewer, Inkview, on Microsoft Windows, and make this the default program for files of type .svg. To do this, you will need to download install the Inkscape package. Inkview.exe is included in the directory you chose for the install, usually C:\program Files\Inkscape. You will need to use the Explorer, Tools, Folder Options, and add extension svg if necessary and then associate svg files with inkview.exe.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

You may also find it convenient to add "c:\program Files\inkscape" to your Path using Control Panel, System, Advanced system settings, Environment variables, Path, and edit to append the folder. You can then use commands like "inkview my_picture.svg" from a command window.

Inkview allows one to open many svg files as a slideshow and move forward and back with the cursor keys.

This works as expected using a command window, for example after changing to the directory containing your SVG files, and running "inkview *.svg" to display all the images in the directory as a slideshow.

Sadly, by default, if you select many files (but not more than 15!) in Windows Explorer, multiple windows are opened instead of a slideshow. If you know how to change this 'by design' feature, please tell me.

You may instead find is useful to create a batch file, for example called demo_plots.bat containing

```
START "Inkview" /MIN inkview.exe  C:\Users\Paul\Desktop\Demo_Plots\*.svg
```

that will display all the svg images in folder \Demo_Plots in turn as the cursor keys are pressed. Of course, you can also omit the folder specification and just have `*.svg` when the location of the batch file will determine which svg files are being viewed.

- Google Chrome
- Many other graphics programs, for example see Most popular SVG software.

The goals of the project are:

- To let users produce simple plots with minimal intervention by using sane defaults.
- To allow users to easily customize plots but allow very fine-grained control of appearance.
- To provide very high quality plots suitable for publication, including printing, but with tiny file sizes.
- To allow the user to talk to the plot classes using coordinate units rather than pixels or other arbitrary units.
- To create the backbone of a `svg` class that can be extended to fully support the standard.
- Compliance with the W3C Scalable Vector Graph standard.

How To Use This Documentation

- Tutorial pages are intended for those who do not know how to use the program.
  - The pages that begin with the tag "Tutorial: Simple" are for absolute beginners.
  - "Tutorial: Full" should give the intermediate user a good reference, as well as show how the program can be used
  - "Tutorial: Special" shows other features/options that the headers provide to the user.
- The pages labelled "**Reference**" give Doxygen reference information. You can find all features with some descriptions, and some examples, in these sections.
- If you have a feature request, or if it appears that the implementation is in error, please check the TODO page first, as well as the rationale and implementation section.

If you do not find your idea/complaint, please reach me either through the Boost development list, or email me direct at pbristow (at) hetp (dot) u-net (dot) com or jakevoytko (at) gmail (dot) com.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Admonishments

Note

These blocks typically go into more detail about an explanation given above.

Tip

These blocks contain information that you may find helpful while coding.

Important

These contain information that is imperative to understanding a concept. Failure to follow suggestions in these blocks will probably result in undesired behavior. Read all of these you find.

Warning

It is imperative that you follow these. Failure to do so will lead to incorrect, and very likely undesired, results in the plot.

Colors

The project supports any RGB color, as well as a number of colors (for example, red, pink, aliceblue...) that are named by the SVG standard.

`svg_color_constant`

`svg_color_constant` is simply an enumerated list of these names colors.

The constants are defined at svg_color.hpp.

Examples of colors are at svg_colors.cpp.

The list of SVG standard named colors contains all the colors you might expect, such as red, blue, green, magenta, cyan, yellow, pink and orange, as well as nearly 150 other shades.

You will probably still find that the colors names do not meet your requirements and so it is also possible to define any RGB combination.

The list also contains one extra color element, `blank`, defined as `false`, used when you need to pass a color, but would not like it to be visible. This comes in handy for defining defaults for functions, for example.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Example of using `svg_color_constant`

```
using namespace boost::svg; // Convenient to use all SVG plot features.

svg_2d_plot my_plot; // Create a 2D plot (with all defaults).

svg_color_constant my_color = red; // Choose a color,
my_plot.background_border_color(my_color); // and apply to the border of the entire image.
my_plot.background_color(lightgray); // Or use directly, to set the color of the background.
```

> **Note**
>
> `svg_color` has a constructor for `svg_color_constant`, so you can use a `svg_color_constant` in place of a `svg_color` and it will be implicitly converted. However, there is **not** currently an `svg_color::operator=(svg_color_constant)` overload, so
>
> ```
> svg_color my_color = red;
> ```
>
> does not have the desired effect, and nor does `cout << red << endl;` output the expected "RGB(255,0,0)" but instead outputs "119", the value of the enum!

`svg_color` interface

You can define a `svg_color` using two different constructors:

```
svg_color(int, int, int); // The parameters are red, green, and blue respectively.

svg_color(svg_color_constant); // Use a pre-existing color constant.
```

Example of using `svg_color`

```
using namespace boost::svg;

svg_color my_white(255, 255, 255); // Using RGB svg_color constructor.
svg_color my_const_white(white); // Using SVG-defined named constructor.

BOOST_ASSERT(my_white == my_const_white);
```

> **Note**
>
> You will find it convenient to use the namespace boost::svg (perhaps only at local scope) to avoid having to fully qualify **every** color that you want to use, for example, to avoid writing: `using boost::svg::azure;`

> **Note**
>
> `svg_color`'s constructor takes in three integer values. The SVG 1.1 standard allows any integer to represent an RGB value, with values less than 0 and greater than 255 being constrained to their respective min and max.

Colors Internals and Rationale

Color constants are defined in an enum, `svg_color_constant`, in alphabetical order at color constants. This facilitates quick lookup of their RGB values from an array. Anywhere that a `svg_color` can be used, a `svg_color_constant` can be used, as the conversion is implicit.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

All color information is stored in RGB format as three 'unsigned char' in a `svg_color` struct. The rationale for storing information in RGB format is that it is precise, and is always representable the exactly the same way. Storing as a floating-point percent or fraction would introduce the possibility of undesirable rounding error.

Fonts

Examples of fonts are at demo_2d_fonts.cpp.

Fine control of font family and size is provided, although the exact rendering may be limited by the browser used to view the .svg file.

An example is provided to demonstrate various fonts and fonts sizes.

1. include
   The conventional wisdom of sticking to one or two fonts are deliberately broken to show various fonts and sizes that are available for SVG plots. The result is a graphic designers nightmare!

   A font family may or may not be available for a particular internet browser, so it is inevitable that the exact appearance of a SVG plot may vary when viewed with different browsers. If a font family is not recognised, then a default (for that browser) will be used instead.

   The font families available for browsers do not seem to be listed, but those used below are available on Mozilla Firefox 3.5. The example below looks very similar with the Adobe SVG Add-in with Microsoft Internet Explorer 8 (which does NOT render SVG natively at all), and with Inkscape and Opera.

   Mozilla Firefox 3.5 allows one to add font(s), but how these are rendered with SVG is not so clear.

   For most purposes the default Font family Verdana looks fine.

   The following font families work with Firefox 3.5:

   arial", "impact", "courier", "lucida console", "Lucida sans unicode", "verdana", "calibri", "century", "lucida calligraphy", "tahoma", "vivaldi" "informal roman", "lucida handwriting", "lucida bright", "helvetica" "arial narrow" is narrow, so may be useful to fit a long title or label. "arial black" is black!

   These do NOT work and are substituted:

   "comic sans", "sans" "bauhaus" "brush script" "segeo condensed" = Serif

   The narrow, wide, bold and italic features produce rather variable and unpredictable results - the rendering may be 'fuzzy' or ill-formed: so these are not recommended. For example,

   "Times New Roman Bold" "Times New Roman Italic" are substituted by "Times New Roman"

   But to get narrow characters "arial narrow" works well, allowing a longer title or label.

   The font sizes are also changes from the defaults. This should change the positioning, but the calculations are complex and necessarily approximate. Collisions between labels, other value labels and axes are not impossible, especially when the tick value labels are not horizontal.

   By default, the std::precision is reduced from the default 6 to 3, and unnecessary zeros and signs are stripped.

   But it will still often be necessary to change the std::iosflags and std::precision, and/or the number of major ticks and/or font size and type to avoid tick value label collisions.

   Unicode symbols can be found at http://en.wikipedia.org/wiki/Unicode_symbols. The 4 hex digit value needs to be wrapped with prefix &#x and suffix ; like &#xDDDD; Rendering of Unicode symbols is not entirely predictable, but usually works well to provide a wide range of greek and math symbols.

2. include

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
The code below shows plotting the sqrt function
```

The code below shows plotting the sqrt function selecting the range of the axis by a user choice.

```
    {
      svg_2d_plot my_plot; // Construct a 2D plot.

      my_plot.legend_on(true)  // Note chaining.
           .title("&#x221A; Function ") // Unicode sqrt symbol.
           .title_font_size(35)
           .title_font_family("arial black")

           .legend_title("Legend title")
           .legend_header_font_size(15)
           .legend_font_family("lucida calligraphy")
           .legend_color(cyan)

           .x_range(0, +20.)
           .x_major_interval(2.)
           .x_num_minor_ticks(4) // MAJOR, minor, minor, minor, minor, MAJOR
           .x_label("x abcd1234")
           .x_axis_label_color(green)
           .x_label_font_family("helvetica")
           .x_label_font_size(40)
           .x_ticks_values_color(red) //
           .x_ticks_values_font_family("Times New Roman")
           .x_ticks_values_font_size(14)
           .x_ticks_values_precision(0)
           .x_ticks_values_ioflags(ios_base::fixed)

           .y_label("sqrt(x) or (&#x221A;x)")
           .y_range(0., 5.)
           .y_ticks_values_color(magenta)
           .y_ticks_values_precision(1)
           .y_ticks_values_ioflags(ios_base::scientific | ios_base::showpos)
           .y_ticks_values_font_family("lucida console")
           .y_ticks_values_font_size(20)
           //.y_label_font_family("informal roman")
           .y_label_font_family("Times New roman")
           .y_label_font_size(40)
           .y_axis_label_color(blue)
         ;

      // Add a container of data to the plot, choosing a color.
     my_plot.plot(data1, "Function (&#x221A;)").stroke_color(red).shape(round).size(3).bezi↵
er_on(true).line_color(pink);

      my_plot.write("./demo_2d_fonts_1.svg"); // Write 1st plot to file.
      // Show some styling, for example for X ticks.
      cout << "my_plot.x_ticks_values_color() " << my_plot.x_ticks_values_color() << endl;
      cout << "my_plot.x_ticks_values_font_family() " << my_plot.x_ticks_values_font_fam↵
ily() << endl;
      cout << "my_plot.x_ticks_values_font_size() " << my_plot.x_ticks_val↵
ues_font_size() << endl;
      cout << "my_plot.x_ticks_values_precision() " << my_plot.x_ticks_values_preci↵
sion() << endl;
      cout << "my_plot.x_ticks_values_ioflags() 0x" << hex << my_plot.x_ticks_values_io↵
flags() << dec << endl;
    }

    // Axis label rotation default is horizontal.
    {
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
      svg_2d_plot my_plot; // Construct another 2D plot to try other variations of font sizes.
      my_plot.x_range(0, +20.)
              .y_range(0., 5.)
              .x_label("X axis label font default size 14")
              .y_label("Y axis label font default size 14")
              //.x_label_font_size(10)
              //.y_label_font_size(10)
              ;
      my_plot.plot(data1, "Function (&#x221A;)").stroke_col↵
or(red).shape(round).size(10).line_on(false).line_color(green);
      my_plot.write("./demo_2d_fonts_1.svg"); // Write another plot to file.
    }

    {
      svg_2d_plot my_plot; // Construct another 2D plot to try other variations of font sizes.
      my_plot.x_range(0, +20.)
              .y_range(0., 5.)
              .x_label("x (small X axis label font size 10)")
              .y_label("y (small X axis label font size 10)")
              .x_label_font_size(10)
              .y_label_font_size(10);
      my_plot.plot(data1, "Function (&#x221A;)").stroke_col↵
or(red).shape(round).size(10).line_on(false).line_color(green);
      my_plot.write("./demo_2d_fonts_2.svg"); // Write another plot to file.
    }

    {
      svg_2d_plot my_plot; // Construct another 2D plot to try other variations of font sizes.
      my_plot.x_range(0, +20.)
              .y_range(0., 5.)
              .x_label("X axis label 30")
              .y_label("Y axis label 30")
             .x_label_font_size(30)
             .y_label_font_size(30);
      my_plot.plot(data1, "Function (&#x221A;)").stroke_col↵
or(red).shape(round).size(10).line_on(false).line_color(green);
      my_plot.write("./demo_2d_fonts_3.svg"); // Write another plot to file.
    }

    {
      svg_2d_plot my_plot; // Construct another 2D plot to try other variations of font sizes.
      my_plot.x_range(0, +20.)
              .y_range(0., 5.)
              .x_label("x (large tick font size 20)")
              .y_label("y (large tick font size 20)")
             .x_label_font_size(10)
             .y_label_font_size(10)
             .x_ticks_values_font_size(20)
             .y_ticks_values_font_size(20)

             ;
      my_plot.plot(data1, "Function (&#x221A;)").stroke_col↵
or(red).shape(round).size(10).line_on(false).line_color(green);
      my_plot.write("./demo_2d_fonts_4.svg"); // Write another plot to file.
    }

// Now alter the rotation of the axis labels.
    {
      svg_2d_plot my_plot; // Construct another 2D plot to try other variations of font sizes.
      my_plot.x_range(0, +20.)
              .y_range(0., 5.)
              .x_label("X axis label font default size 14")
              .y_label("Y axis label font default size 14")
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
            .x_major_label_rotation(uphill)
            .y_major_label_rotation(uphill)
            //.x_label_font_size(10)
            //.y_label_font_size(10)
            ;
      my_plot.plot(data1, "Function (&#x221A;)").stroke_col↵
or(red).shape(round).size(10).line_on(false).line_color(green);
      my_plot.write("./demo_2d_fonts_5.svg"); // Write another plot to file.
    }

    {
     svg_2d_plot my_plot; // Construct another 2D plot to try other variations of font sizes.
      my_plot.x_range(0, +20.)
            .y_range(0., 5.)
            .x_label("x (small X axis label font size 10)")
            .y_label("y (small X axis label font size 10)")
            .x_label_font_size(10)
            .y_label_font_size(10)
            .x_major_label_rotation(uphill)
            .y_major_label_rotation(uphill)
            ;

      my_plot.plot(data1, "Function (&#x221A;)").stroke_col↵
or(red).shape(round).size(10).line_on(false).line_color(green);
      my_plot.write("./demo_2d_fonts_6.svg"); // Write another plot to file.
    }

    {
     svg_2d_plot my_plot; // Construct another 2D plot to try other variations of font sizes.
      my_plot.x_range(0, +20.)
            .y_range(0., 5.)
            .x_label("X axis label 30")
            .y_label("Y axis label 30")
            .x_label_font_size(30)
            .y_label_font_size(30)
            .x_major_label_rotation(uphill)
            .y_major_label_rotation(uphill)
            ;
      my_plot.plot(data1, "Function (&#x221A;)").stroke_col↵
or(red).shape(round).size(10).line_on(false).line_color(green);
      my_plot.write("./demo_2d_fonts_7.svg"); // Write another plot to file.
    }

    {
     svg_2d_plot my_plot; // Construct another 2D plot to try other variations of font sizes.
      my_plot.x_range(0, +20.)
            .y_range(0., 5.)
            .x_label("x tick size 12, label 14")
            .y_label("y tick size 12, label 14")
           .x_label_font_size(14)
           .y_label_font_size(14)
           .x_ticks_values_font_size(12)
           .y_ticks_values_font_size(12)
            .x_major_label_rotation(uphill)
            .y_major_label_rotation(uphill)
           ;
      my_plot.plot(data1, "Function (&#x221A;)").stroke_col↵
or(red).shape(round).size(10).line_on(false).line_color(green);
      my_plot.write("./demo_2d_fonts_8.svg"); // Write another plot to file.
    }

    {
     svg_2d_plot my_plot; // Construct another 2D plot to try other variations of font sizes.
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
      my_plot.x_range(0, +20.)
              .y_range(0., 5.)
              .x_label("X axis label font default size 14")
              .y_label("Y axis label font default size 14")
              .x_major_label_rotation(downward)
              .y_major_label_rotation(upward)
              ;
      my_plot.plot(data1, "Function (&#x221A;)").stroke_col↵
or(red).shape(round).size(10).line_on(false).line_color(green);
      my_plot.write("./demo_2d_fonts_9.svg"); // Write another plot to file.
    }

    {
     svg_2d_plot my_plot; // Construct another 2D plot to try other variations of font sizes.
      my_plot.x_range(0, +20.)
              .y_range(0., 5.)
              .x_label("x (small X axis label font size 10)")
              .y_label("y (small X axis label font size 10)")
              .x_label_font_size(10)
              .y_label_font_size(10)
              .x_major_label_rotation(steepdown)
              .y_major_label_rotation(steepup)
              ;

      my_plot.plot(data1, "Function (&#x221A;)").stroke_col↵
or(red).shape(round).size(10).line_on(false).line_color(green);
      my_plot.write("./demo_2d_fonts_10.svg"); // Write another plot to file.
    }

    {
     svg_2d_plot my_plot; // Construct another 2D plot to try other variations of font sizes.
      my_plot.x_range(0, +20.)
              .y_range(0., 5.)
              .x_label("X axis label 30")
              .y_label("Y axis label 30")
              .x_label_font_size(30)
              .y_label_font_size(30)
              .x_major_label_rotation(downhill)
              .y_major_label_rotation(uphill)
              ;
      my_plot.plot(data1, "Function (&#x221A;)").stroke_col↵
or(red).shape(round).size(10).line_on(false).line_color(green);
      my_plot.write("./demo_2d_fonts_11.svg"); // Write another plot to file.
    }

    {
     svg_2d_plot my_plot; // Construct another 2D plot to try other variations of font sizes.
      my_plot.x_range(0, +20.)
              .y_range(0., 5.)
              .x_label("x tick size 12, label 14")
              .y_label("y tick size 12, label 14")
             .x_label_font_size(14)
             .y_label_font_size(14)
             .x_ticks_values_font_size(12)
             .y_ticks_values_font_size(12)
              .x_major_label_rotation(slopedownhill)
              .y_major_label_rotation(slopeup)
             ;
      my_plot.plot(data1, "Function (&#x221A;)").stroke_col↵
or(red).shape(round).size(10).line_on(false).line_color(green);
      my_plot.write("./demo_2d_fonts_12.svg"); // Write another plot to file.
    }
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
  }
  catch(const std::exception& e)
  {
    std::cout <<
      "\n""Message from thrown exception was:\n   " << e.what() << std::endl;
  }

  return 0;
} // int main()
```

Producing this ugly plot, as an example from the files demo_2d_fonts_1.svg to demo_2d_fonts_12.svg produce by this example.

See demo_2d_fonts.cpp for full source code.

1D Tutorials

This section gives examples of plotting 1-dimensional data held in, for example, in an array or vector.

1-D Vector Example

First we need a few includes to use Boost.Plot:

```
#include <boost/svg_plot/svg_1d_plot.hpp>
  using namespace boost::svg;
#include <vector>
  using std::vector;
```

STL vector is used as the container for our two data series, and values are inserted using push_back. Since this is a 1-D plot the order of data values is not important.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
vector<double> dan_times;
dan_times.push_back(3.1);
dan_times.push_back(4.2);

vector<double> elaine_times;
elaine_times.push_back(2.1);
elaine_times.push_back(7.8);
```

The constructor initializes a new 1D plot, called `my_plot`, and also sets all the very many defaults for axes, width, colors, etc.

```
svg_1d_plot my_plot;
```

A few (member) functions that set are fairly self-explanatory:

- title provides a title at the top for the whole plot,
- `legend_on(true)` will mean that titles of data series and markers will display in the legend box.
- `x_range(-1, 11)` sets the axis limits from -1 to +11 (instead of the default -10 to +10).
- `background_border_color(blue)` sets just one of the very many options.

```
my_plot.background_border_color(blue)
  .legend_on(true)
  .title("Race Times")
  .x_range(-1, 11);

my_plot.legend_lines(true);
```

The syntax `my_plot.title("Hello").legend_on(true)...` may appear unfamiliar, but is a convenient way of specifying many parameters in any order. It is equivalent to:

```
my_plot.title("Race Times");
my_plot.legend_on(true);
my_plot.x_range(-1, 11);
my_plot.background_border_color(blue);
```

Chaining thus allows you to avoid repeatedly typing "`myplot.`" and easily group related settings like plot window, axes ... together. A fixed order would clearly become impracticable with hundreds of possible arguments needed to set all the myriad plot options.

Within all of the plot classes, 'chaining' works the same way, by returning a reference to the calling object thus `return *this;`

Then we need to add our data series, and add optional (but very helpful) data series titles if we want them to show on the legend.

```
my_plot.plot(dan_times, "Dan").shape(round).size(10).stroke_color(red).fill_color(green);
my_plot.plot(elaine_times, "Elaine").shape(vertical_line).stroke_color(blue);
```

Finally, we can write the SVG to a file of our choice.

```
my_plot.write("./demo_1d_vector.svg");
```

The IDE output is not very exciting in this case

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
Compiling...
demo_1d_vector.cpp
Linking...
Embedding manifest...
Autorun "j:\Cpp\SVG\debug\demo_1d_vector.exe"
Build Time 0:04
```

but the plot is

(You can also view the SVG XML source using browsers (View, Source) or use your favorite text editor like Notepad, TextPad etc at demo_1d_vector.svg

See demo_1d_vector.cpp for full source code.

1-D STL Containers Examples

First a few includes to use Boost.Plot and various STL containers:

```
#include <boost/svg_plot/svg_1d_plot.hpp>
  using namespace boost::svg;
#include <boost/array.hpp>
  using boost::array;
#include <vector>
  using std::vector;
#include <set>
  using std::set; // Automatically sorted - though this is not useful to the plot process.
  using std::multiset; // At present using multiset, allowing duplicates, plot does not indic↵
ate duplicates.
  // With 2_D data in multimap, duplicate values are usefully displayed.
#include <list>
  using std::list;
#include <deque>
  using std::deque;
```

STL vector is used as the container, and fictional values are inserted using push_back. Since this is a 1-D plot the order of data values is not important.

```
vector<float> values;
values.push_back(3.1f);
values.push_back(-5.5f);
values.push_back(8.7f);
values.push_back(0.5f);
```

The constructor initializes a new 1D plot, called `my_plot`, and also sets all the many default values.

```
svg_1d_plot my_plot;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Setting (member) functions are fairly self-explanatory: Title provides a title at the top for the whole plot, and plot adds a (unamed) data series (naming isn't very useful if there is only one data series).

```
my_plot.title("vector&lt;float&gt; example");
```

> **Note**
>
> One must insert the XML character entity equivalents of < for < and > for >).

```
my_plot.plot(values);
```

Write the SVG to a file.

```
my_plot.write("./demo_1d_vector_float.svg");
```

If the container is a static array, then it must be filled by assignment:

```
array<long double, 4> values = {3.1L,-5.5L, 8.7L, 0.5L};

svg_1d_plot my_plot;
my_plot.title("array&lt;long double&gt; example");
my_plot.plot(values);
my_plot.write("./demo_1d_array_long_double.svg");
```

If the container type is a set, then it can be filled with insert:

```
set<double> values;
values.insert(-8.4);
values.insert(-2.3);
values.insert(0.1);
values.insert(5.6);
values.insert(7.8);

svg_1d_plot my_plot;
my_plot.title("set&lt;double&gt; example");
my_plot.plot(values);
my_plot.write("./demo_1d_set_double.svg");
```

If the container type is a list, then it can be filled with push_back or push_front:

```
list<double> values;
values.push_back(-8.4);
values.push_back(-2.3);
values.push_back(0.1);
values.push_back(5.6);
values.push_back(7.8);
svg_1d_plot my_plot;
my_plot.title("list&lt;double&gt; example");
my_plot.plot(values);
my_plot.write("./demo_1d_list_double.svg");
```

If the container type is a deque, then it can be filled with push_back or push_front:

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
deque<double> values;
values.push_front(-8.4);
values.push_front(-2.3);
values.push_front(0.1);
values.push_front(5.6);
values.push_front(7.8);

svg_1d_plot my_plot;
my_plot.title("deque&lt;double&gt; example");
my_plot.plot(values);
my_plot.x_label("X values as doubles");

my_plot.write("./demo_1d_deque_double.svg");
```

Note

For filling STL containers you may find the Boost.Assign library by Thorsten Ottosen useful.

The IDE output is not very exciting in this case:

```
Compiling...
demo_1d_containers.cpp
Linking...
Embedding manifest...
Autorun "j:\Cpp\SVG\debug\demo_1d_containers.exe"
Build Time 0:03
```

The plot is:

And you can view the other svg files (with most internet browsers, and other programs too) for example:

- demo_1d_array_long_double.svg
- demo_1d_set_double.svg
- demo_1d_list_double.svg
- demo_1d_deque_double.svg

See demo_1d_containers.cpp for full source code.

Tutorial: 1D Autoscale with Multiple Containers

This example demonstrates autoscaling with **multiple** STL containers.

First we need a few includes to use Boost.Plot (and some others only needed for this example).

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
#include <boost/svg_plot/svg_1d_plot.hpp>
  using namespace boost::svg;
  using boost::svg::svg_1d_plot;

#include <boost/svg_plot/detail/auto_axes.hpp>
  using boost::svg::show; // A single STL container.
  using boost::svg::show_all; // Multiple STL containers.
  using boost::svg::range; // Find min and max of a single STL container.
  using boost::svg::range_all;// Find min and max of multiple STL containers.
  // Note neither of these check for 'limits' (infinite, NaN) values.

#include <boost/svg_plot/detail/pair.hpp>
  using boost::svg::detail::operator<<; // Output pair as, for example: 1.23, 4.56

//#include <boost/svg_plot/show_1d_settings.hpp>
// Only needed for showing which settings in use.
//void boost::svg::show_1d_plot_settings(svg_1d_plot&);

#include <boost/algorithm/minmax.hpp>
 using boost::minmax;
#include <boost/algorithm/minmax_element.hpp>
 using boost::minmax_element;

#include <iostream>
  using std::cout;
  using std::endl;
  using std::boolalpha;

#include <limits>
  using std::numeric_limits;

#include <vector>
  using std::vector;

#include <utility>
  using std::pair;

#include <algorithm>
  using std::multiplies;
  using std::transform;
  using std::copy;
#include <iterator>
  using std::iterator;
  using std::iterator_traits;
  using std::ostream_iterator;
```

This example uses two containers to demonstrate autoscaling. It is common to plot more than one set of data series together. Autoscaling must probably inspect all the containers of these series in order to find axis ranges that will be suitable for all of them.

```
vector<double> my_data_1;
// Initialize STL container my_data_1 with some entirely fictional data.
my_data_1.push_back(0.2); // [0]
my_data_1.push_back(1.1); // [1]
my_data_1.push_back(4.2); // [2]
my_data_1.push_back(3.3); // [3]
my_data_1.push_back(5.4); // [4]
my_data_1.push_back(6.5); // [5]
```

We might use some convenient functions to list the container to cout.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
show(my_data_1); // Entire container contents.
```

Others are easily written, often using std::copy, for example:

```
copy(my_data_1_2.begin(), my_data_1_2.end(), ostream_iterator<double>(cout, " "));
```

Now we concoct another equally fictional data series by a transform multiplying by 2.3.

```
vector<double> my_data_2; // Create a second data series.
copy(my_data_1.begin(), my_data_1.end(), back_inserter(my_data_2));
// Change the values in an entirely arbitrary way (each * 2.3).
transform(my_data_2.begin(), my_data_2.end(), my_data_2.begin(), bind1st(multi↵
plies<double>(), 2.3));
//cout << endl << my_data.size() << " values in my_data_2. " << endl;
```

Next we need a new STL container, vector say, to hold our multiple containers of data series. They must all be the same STL container type, in this example, vector<double>. And we use pushback to add the containers.

```
vector<vector<double> > my_containers;

my_containers.push_back(my_data_1); // Add 1st data series.
my_containers.push_back(my_data_2); // Add another data series.
cout << my_containers.size() << " containers." << endl;
show_all(my_containers);
```

Finally we can use all the containers to find the minimum of mimimums and maximum of maximums ready to feed into the plot autoscale function.

```
pair<double, double> mm = range_all(my_containers);
cout << "Data range: "  << mm << endl; // min & max of all data.
svg_1d_plot my_1d_plot; // Construct a plot with all the default constructor values.
```

We could feed the minimum and maximum values separately,

```
my_1d_plot.x_autoscale(mm.first, mm.second);  // Use minimum and maximum to autoscale.
```

but usually feeding the pair is more convenient. (Perhaps we might want to impose some other minimum and maximum here).

```
my_1d_plot.x_autoscale(mm);  // Use overall minimum and maximum to autoscale.
```

Finally, we add the data series containers to the plot, and write the SVG out to file.

```
my_1d_plot.plot(my_data_1, "data_1");
my_1d_plot.plot(my_data_2, "data_2").stroke_color(red);

my_1d_plot.write("auto_1d_containers.svg"); // Write the plot to file.
```

If we want, we can check the autoscale range used.

```
cout << "x_range() " << my_1d_plot.x_range() << endl; // x_range() 0, 15
```

And even all the (hundreds of) plot settings (useful for diagnosis why your plot doesn't meet your expectations).

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
//show_1d_plot_settings(my_1d_plot);
```

Warning

The containers must be of the same type to use the function range_all. If different types of containers, for example some in a set and some in a vector, then the min and max for each container must be computed separately and the minimum of the minimums and the maximum of the maximums injected into the x_autoscale (and/or y_autoscale) call.

Typical output is:

```
Compiling...
auto_1d_containers.cpp
Linking...
Embedding manifest...
Autorun "j:\Cpp\SVG\debug\auto_1d_containers.exe"
6 values in container: 0.2 1.1 4.2 3.3 5.4 6.5
2 containers.
6 values in container: 0.2 1.1 4.2 3.3 5.4 6.5
6 values in container: 0.46 2.53 9.66 7.59 12.42 14.95
Data range: 0.2, 14.9
x_range() 0, 15
Build Time 0:03
```

See auto_1d_containers.cpp for full source code and sample output.

Tutorial: 1D More Layout Examples

This section showsa few more of the near infinite layout options. See also the reference section and function index.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
#include <boost/svg_plot/svg_1d_plot.hpp>
using namespace boost::svg;
#include <cmath>  // for sqrt

#include <vector>
#include <deque>
#include <boost/array.hpp>

// Three different STL-style containers:
using std::vector;
using std::deque;
using boost::array;

// Three sample functions:
double f(double x)
{
  return sqrt(x);
}

double g(double x)
{
  return -2 + x*x;
}

double h(double x)
{
  return -1 + 2*x;
}

int main()
{
  vector<double> data1;
  deque<double> data2;
  boost::array<double, 10> data3;
  // Fill the containers with some data using the functions:
  int j=0;
  for(double i=0; i<9.5; i+=1.)
  {
    data1.push_back(f(i));
    data2.push_front(g(i));
    data3[j++] = h(i);
  }

  svg_1d_plot my_plot; // Construct my_plot.

  // Set many plot options,
  // using chaining to group similar aspects together.
  // Size/scale settings.
  my_plot.image_size(500, 350)
         .x_range(-3, 10);

  // Text settings.
  my_plot.title("Oh My!")
         .title_font_size(29)
         .x_label("Time in Months") // X- axis label
         .x_label_on(true); // Not needed, because although the default is no axis label,
         // providing a label has this effect.
         // If we want later to switch *off* labels,
         // my_plot.x_label_on(false); // is needed.

  // Commands.
  my_plot.legend_on(true)
         .plot_window_on(true)
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
        .x_major_labels_on(true);

  // Color settings.
  my_plot.background_color(svg_color(67, 111, 69))
        .legend_background_color(svg_color(207, 202,167))
        .legend_border_color(svg_color(102, 102, 84))
        .plot_background_color(svg_color(136, 188, 126))
        .title_color(white);

  // Axis settings.
  my_plot.x_major_interval(2)
        .x_major_tick_length(14)
        .x_major_tick_width(1)
        .x_minor_tick_length(7)
        .x_minor_tick_width(1)
        .x_num_minor_ticks(3);

  // Legend settings.
  my_plot.legend_title_font_size(15);

  // Put the three containers of data onto the plot.
  my_plot.plot(data1, "Lions",  blue);
  my_plot.plot(data2, "Tigers", orange);
  my_plot.plot(data3, "Bears",  red);

  my_plot.write("1d_full_layout.svg");

  return 0;
} // int main()
```

This produces the following plot:

A little bit of color customization goes a **very** long way!

Tutorial: 1D Gridlines & Axes - more examples

It is possible to change the location of the intersection of the axes, and the labelling - at the top, bottom or in the middle on the Y axis. This is controlled using the `enum x_axis_intersect` and function `x_ticks_on_window_or_axis`.

```
int main()
{
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Following previous examples, we set up two containers for two data series.

```
vector<double> dan_times;
vector<double> elaine_times;

dan_times.push_back(3.1);
dan_times.push_back(4.2);
elaine_times.push_back(2.1);
elaine_times.push_back(7.8);

svg_1d_plot my_plot;

// Adding some generic settings.
my_plot.background_border_color(black)
       .legend_on(true)
       .plot_window_on(true)
       .title("Race Times")
       .x_range(-1, 10);
```

We add tastelessly color the grids for both major and minor ticks, and switch both grids on.

```
my_plot.x_major_grid_color(pink)
       .x_minor_grid_color(lightgray);

my_plot.x_major_grid_on(true)
       .x_minor_grid_on(true);
```

Also we specify the position of the labelling of the X-axis. It can be controlled using values in the enum x_axis_intersect.

```
enum x_axis_intersect
{ //! \enum x_axis_intersect
  bottom = -1, // On the bottom of the plot window.
  x_intersects_y = 0, // On the Y axis (in the middle of the plot window).
  top = +1 // On the top of the plot window.
};
```

For this example, we choose to show the X axis and tick value labels at the top of the plot window.

```
  my_plot.x_ticks_on_window_or_axis(top); // on top, not on axis.

  // Write to plot.
  my_plot.plot(dan_times, "Dan").stroke_color(blue);
  my_plot.plot(elaine_times, "Elaine").stroke_color(orange);

  // Write to file.
  my_plot.write("./demo_1d_x_external.svg");
  return 0;
} // int main()
```

X-Axis Grid Lines

If you would like vertical grid lines that go on the graph, you can make the following call to `svg_1d_plot`:

```
my_plot.x_major_grid_on(true)
       .x_minor_grid_on(true);
```

To color style it, you might add the following calls:

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
my_plot.x_major_grid_color(lightgray) // Darker color for major grid.
       .x_minor_grid_color(whitesmoke); // Lighter color for minor grid.
```

This will produce the following image:

X-Axis Tick value labels precision, iosflags, font family and size

An example to demonstrate some options for controlling the layout and color of tick values.

```
The code below shows plotting the sqrt function
```

The code below shows plotting the sqrt function selecting the range of the axis by a user choice.

Note

Unicode symbols can be found at http://en.wikipedia.org/wiki/Unicode_symbols. The 4 hex digit value needs to be wrapped with prefix &#x and suffix ; like &#xDDDD;

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
svg_2d_plot my_plot; // Construct a 2D plot.

my_plot.legend_on(true) // Set title and legend, and X and Y axis range.
       .title("&#x221A; Function ") // Unicode sqrt symbol.
       .x_range(0, +20.)
       .x_major_interval(2.)

       .x_axis_label_color(green)
       .x_label_font_family("helvetica")
       .x_label_font_size(30)

       .x_num_minor_ticks(4) // MAJOR, minor, minor, minor, minor, MAJOR
       .x_ticks_values_color(red) //
       .x_ticks_values_font_family("Times New Roman")
       .x_ticks_values_font_size(20)
       .x_ticks_values_precision(0)
       .x_ticks_values_ioflags(ios_base::fixed)

       .y_range(0., 5.)
       .y_ticks_values_color(magenta)
       .y_ticks_values_precision(1)
       .y_ticks_values_ioflags(ios_base::scientific | ios_base::showpos)

        // "arial", "impact", "courier", "lucida console",  "Lucida sans unicode", "verdana", ↵
"calibri", "century", "lucida calligraphy", "tahoma", "vivaldi"
        // "informal roman", "lucida handwriting", "lucida bright", "helvetica"
        // "arial narrow" is narrow, so may be useful.
        // "arial black" is black!
        // "Times New Roman Bold" "Times New Roman italic" = Times New Roman
        // "comic sans", "sans" "bauhaus" "brush script" "segeo condensed" = Serif

        .y_ticks_values_font_family("lucida console")
        .y_ticks_values_font_size(10)

        .y_label_font_family("Times New Roman")
        .y_label_font_size(30)
        .y_axis_label_color(blue)
      ;

my_plot.x_label("x abcd1234(&#x221A;)").y_label("sqrt(x) "); // Note chaining.

// Add a container of data to the plot, choosing a color.
my_plot.plot(data1, "Function (&#x221A;)").stroke_color(red).shape(round).size(3).bezi↵
er_on(true).line_color(pink);
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
    my_plot.write("./demo_2d_tick_values.svg"); // Write the plot to another file.

    // Show the ticks styling:
    // X ticks.
    cout << "my_plot.x_ticks_values_color() " << my_plot.x_ticks_values_color() << endl;
    cout << "my_plot.x_ticks_values_font_family() " << my_plot.x_ticks_values_fam↵
ily() << endl;
    cout << "my_plot.x_ticks_values_font_size() " << my_plot.x_ticks_values_font_size() << endl;
    cout << "my_plot.x_ticks_values_precision() " << my_plot.x_ticks_values_precision() << endl;
    cout << "my_plot.x_ticks_values_ioflags() 0x" << hex << my_plot.x_ticks_values_io↵
flags() << dec << endl;
    // Y ticks.
    cout << "my_plot.y_ticks_values_color() " << my_plot.y_ticks_values_color() << endl;
    cout << "my_plot.y_ticks_values_font_family() " << my_plot.y_ticks_values_fam↵
ily() << endl;
    cout << "my_plot.y_ticks_values_font_size() " << my_plot.y_ticks_values_font_size() << endl;
    cout << "my_plot.y_ticks_values_color() " << my_plot.y_ticks_values_color() << endl;
    cout << "my_plot.y_ticks_values_precision() " << my_plot.y_ticks_values_precision() << endl;
    cout << "my_plot.y_ticks_values_ioflags() 0x" << hex << my_plot.y_ticks_values_io↵
flags() << dec << endl;

  }
  catch(const std::exception& e)
  {
    std::cout <<
      "\n""Message from thrown exception was:\n   " << e.what() << std::endl;
  }

  return 0;
} // int main()

//    cout << "my_plot.x_ticks_values_color() " << my_plot.x_ticks_values_color() << endl;
```

producing this plot

See demo_2d_tick_values.cpp for full source code.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

1-D Axis Scaling

Axis scaling with function scale_axis

This example shows the use of functions scale_axis to find suitable axis limits. Normally one would use autoscaling, but there are conceivable circumstances when one would want to check on the algorithms choice of axis and perhaps intervene in the process.

First some includes to use Boost.Plot (and some others only needed for this example).

```
#include <boost/svg_plot/svg_1d_plot.hpp>
  using namespace boost::svg;
  using boost::svg::svg_1d_plot;

#include <boost/svg_plot/show_1d_settings.hpp>
// Only needed for showing which settings in use.
 void boost::svg::show_1d_plot_settings(svg_1d_plot&);
 // (Also provides operator<< for std::pair).

#include <boost/algorithm/minmax.hpp>
 using boost::minmax;
#include <boost/algorithm/minmax_element.hpp>
 using boost::minmax_element;

#include <iostream> // for debugging.
  using std::cout;
  using std::endl;
  using std::boolalpha;

#include <limits>
  using std::numeric_limits;

#include <vector>
  using std::vector;
#include <set>
  using std::multiset;

#include <utility>
  using std::pair;

#include <boost/svg_plot/detail/auto_axes.hpp>
  using boost::svg::show; // A single STL container.
  using boost::svg::show_all; // Multiple STL containers.
  using boost::svg::range; // Find min and max of a STL container.
  using boost::svg::range_all;// Find min and max of multipler STL containers.
```

This example uses a few types of containers to demonstrate axis_scaling. axis_scaling must inspect the container in order to find axis ranges that will be suitable. First we create a container and fill with some fictional data.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
vector<double> my_data;
// Initialize my_data with some entirely fictional data.
my_data.push_back(0.2);
my_data.push_back(1.1); // [1]
my_data.push_back(4.2); // [2]
my_data.push_back(3.3); // [3]
my_data.push_back(5.4); // [4]
my_data.push_back(6.5); // [5]
show(my_data); // Show entire container contents,
// 6 values in container: 0.2 1.1 4.2 3.3 5.4 6.5
```

```
multiset<double> my_set;
// Initialize my_set with some entirely fictional data.
my_set.insert(1.2);
my_set.insert(2.3);
my_set.insert(3.4);
my_set.insert(4.5);
my_set.insert(5.6);
my_set.insert(6.7);
my_set.insert(7.8);
my_set.insert(8.9);
// Show the set.
multiset<double>::const_iterator si;
show(my_set); // for two different types of container.
// 8 values in container: 1.2 2.3 3.4 4.5 5.6 6.7 7.8 8.9
```

Show can also display just a part of the container contents.

```
// show(&my_data[0], &my_data[my_data.size()]); // pointers - wrong! > all data ;-)
show(&my_data[0], &my_data[my_data.size()-1]); // pointers, all data.
show(&my_data[1], &my_data[5]); // pointers, part data.
show(my_data.begin(), my_data.end()); // iterators.
show(++(my_data.begin()), --(my_data.end())); // Just the 4 middle values.

vector<double>::const_iterator idb = my_data.begin();
vector<double>::const_iterator ide = my_data.end();
show(idb, ide); // All
++idb; // Move to 2nd value.
--ide; // move back from last value.
show(idb, ide); // Just the 4 middle values.
```

scale_axis Function Examples

Is is possible to find the minimum and maximum values in a container using the STL functions min & max or more conveiniently and efficiently with boost::minmax_element:

```
typedef vector<double>::const_iterator vector_iterator;
pair<vector_iterator, vector_iterator> result = boost::minmax_element(my_data.be↵
gin(), my_data.end());
cout << "The smallest element is " << *(result.first) << endl; // 0.2
cout << "The largest element is  " << *(result.second) << endl; // 6.5

// axis_scaling using two double min and max values.
double min_value = *(my_data.begin());
double max_value = *(--my_data.end());
cout << "axis_scaling 1 min " << min_value << ", max = " << max_value << endl;
```

and to apply these values to the axis_scaling algorithm using by plot to choose the axes limits and ticks.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
double axis_min_value; // Values to be updated by function scale_axis.
double axis_max_value;
double axis_tick_increment;
int axis_ticks;

scale_axis(min_value, max_value,
  &axis_min_value, &axis_max_value, &axis_tick_increment, &axis_ticks,
  false, tol100eps, 6); // Display range.
cout << "Axis_scaled 2 min " << axis_min_value << ", max = " << axis_max_value << ", increment ↵
" << axis_tick_increment << endl;
```

It is also possible to use this with containers that use iterators and whose contents are ordered in ascending value, axis_scaling using first and last in container, for example, set, map, multimap, or a sorted vector or array. A number of variations are shown below, mianly by way of testing.

```
scale_axis(*my_data.begin(),*(--my_data.end()),
  &axis_min_value, &axis_max_value, &axis_tick_increment, &axis_ticks,
  false, tol100eps, 6); // Display range.
cout << "Axis_scaled 3 min " << axis_min_value << ", max = " << axis_max_value << ", increment ↵
" << axis_tick_increment << endl;

// axis_scaling using two begin & end iterators into STL container,
// scale_axis does finding min and max.
scale_axis(my_data.begin(), my_data.end(), // Input range
  &axis_min_value, &axis_max_value, &axis_tick_increment, &axis_ticks, // to update.
  true, // check for non-finite
  3., // autoscale_plusminus = 3 sd
  false, // Do not include origin
  tol100eps, // tight
  6, // steps at default.
  0); // Display range.
cout << "Axis_scaled 4 min " << axis_min_value << ", max = " << axis_max_value << ", increment ↵
" << axis_tick_increment << endl;

// axis_scaling using two begin & end iterators into STL container,
// scale_axis does finding min and max.
scale_axis(my_data[1], my_data[4], // Only middle part of the container used, ignoring 1st and ↵
last values.
  &axis_min_value, &axis_max_value, &axis_tick_increment, &axis_ticks,
  true, tol100eps, 6); // Display range.
cout << "Axis_scaled 5 min " << axis_min_value << ", max = " << axis_max_value << ", increment ↵
" << axis_tick_increment << endl;

// axis_scaling using whole STL container,
// scale_axis does finding min and max.
scale_axis(my_data, &axis_min_value, &axis_max_value, &axis_tick_increment, &axis_ticks,
  true, 3., false, tol100eps, 6); // Display range.
cout << "Axis_scaled 6 min " << axis_min_value << ", max = " << axis_max_value << ", increment ↵
" << axis_tick_increment << endl;

svg_1d_plot my_1d_plot; // Construct a plot with all the default constructor values.

// One could intercept and change any values calculated by scale_axis here?
// Set the plot to use range and interval from the scale_axis values.
```

The axis range thus computed can be inserted directly into the plot using the `range` and `x_major_interval` functions.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
my_1d_plot.x_range(axis_min_value, axis_max_value)
  .x_major_interval(axis_tick_increment);

//my_1d_plot.x_autoscale(false);  // Ensure autoscale values are *not* recalculated for the plot.
```

```
// Set some axis_scaling parameters:
my_1d_plot.x_with_zero(false);
my_1d_plot.x_min_ticks(10);
my_1d_plot.x_steps(0);
my_1d_plot.x_tight(0.001);

// Show the flags just set.
cout << (my_1d_plot.x_with_zero() ? "x_with_zero, " : "not x_with_zero, ")
  << my_1d_plot.x_min_ticks() << " x_min_ticks, "
  << my_1d_plot.x_steps() << " x_steps, "
  << my_1d_plot.x_tight() << " tightness." << endl;

my_1d_plot.x_autoscale(my_data);  // Use all my_data to autoscale.
cout << "Axis_scaled " // Show the results of autoscale:
  "min " << my_1d_plot.x_auto_min_value()
  << ", max "<< my_1d_plot.x_auto_max_value()
  << ", interval " << my_1d_plot.x_auto_tick_interval() << endl; // Autoscaled min 0, max 6.5, ↵
interval 0.5

my_1d_plot.x_autoscale(my_data.begin(), my_data.end());  // Use all my_data to autoscale.

cout << "Axis_scaled " // Show the results of autoscale:
  "min " << my_1d_plot.x_auto_min_value()
  << ", max "<< my_1d_plot.x_auto_max_value()
  << ", interval " << my_1d_plot.x_auto_tick_interval() << endl; // Autoscaled min 0, max 6.5, ↵
interval 0.5

my_1d_plot.x_autoscale(my_data[1], my_data[4]);  // Use only part of my_data to autoscale.

cout << "Axis_scaled " // Show the results of autoscale:
  "min " << my_1d_plot.x_auto_min_value()
  << ", max "<< my_1d_plot.x_auto_max_value()
  << ", interval " << my_1d_plot.x_auto_tick_interval() << endl; // Autoscaled min 1, max 5.5, ↵
interval 0.5

my_1d_plot.x_autoscale(true);  // Ensure autoscale values are used for the plot.

my_1d_plot.plot(my_data, "Auto 1D"); // Add the one data series.
cout << "Axis_scaled " // Show the results of autoscale:
  " min " << my_1d_plot.x_auto_min_value()
  << ", max "<< my_1d_plot.x_auto_max_value()
  << ", interval " << my_1d_plot.x_auto_tick_interval() << endl; // Autoscaled min 1, max 5.5, ↵
interval 0.5

my_1d_plot.plot(my_set.begin(), my_set.end(), "Auto 1D"); // Add another data series from my_set.
my_1d_plot.plot(my_set, "Auto 1D"); // Add another whole data series from my_set.
my_1d_plot.plot(&my_data[1], &my_data[4], "Auto 1D"); // Add part (1,2 3 but *not* 4) of the ↵
one data series.
//my_1d_plot.plot(&my_set[1], &my_set[4], "Auto 1D"); // operator[] is not defined for set con↵
tainer!
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
my_1d_plot.write("demo_1d_axis_scaling.svg"); // Write the plot to file.

using boost::svg::detail::operator<<; // Needed for output a pair.
cout << "x_range() " << my_1d_plot.x_range() << endl; // x_range() 1, 5.5

//show_1d_plot_settings(my_1d_plot);
```

The output is:

```
Autorun "j:\Cpp\SVG\Debug\demo_1d_axis_scaling.exe"
6 values in container: 0.2 1.1 4.2 3.3 5.4 6.5
8 values in container: 1.2 2.3 3.4 4.5 5.6 6.7 7.8 8.9
0.2 1.1 4.2 3.3 5.4 : 5 values used.
1.1 4.2 3.3 5.4 : 4 values used.
0.2 1.1 4.2 3.3 5.4 6.5 : 6 values used.
1.1 4.2 3.3 5.4 : 4 values used.
0.2 1.1 4.2 3.3 5.4 6.5 : 6 values used.
1.1 4.2 3.3 5.4 : 4 values used.
The smallest element is 0.2
The largest element is  6.5
axis_scaling 1 min 0.2, max = 6.5
Axis_scaled 2 min 0, max = 7, increment 1
Axis_scaled 3 min 0, max = 7, increment 1
Axis_scaled 4 min 0, max = 7, increment 1
Axis_scaled 5 min 0, max = 6, increment 1
Axis_scaled 6 min 0, max = 7, increment 1
not x_with_zero, 10 x_min_ticks, 0 x_steps, 0.001 tightness.
Axis_scaled min 0, max 6.5, interval 0.5
Axis_scaled min 0, max 6.5, interval 0.5
Axis_scaled min 1, max 5.5, interval 0.5
Axis_scaled  min 1, max 5.5, interval 0.5
x_range() 1, 5.5
```

See demo_1d_axis_scaling.cpp for full source code.

1-D Auto scaling Examples

First we need a few includes to use Boost.Plot (and some others only needed for this example).

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
#include <boost/svg_plot/svg_1d_plot.hpp>
  using namespace boost::svg;
  using boost::svg::svg_1d_plot;

#include <boost/svg_plot/show_1d_settings.hpp>
// Only needed for showing which settings in use.
 void boost::svg::show_1d_plot_settings(svg_1d_plot&);
 // (Also provides operator<< for std::pair).

#include <boost/algorithm/minmax.hpp>
 using boost::minmax;
#include <boost/algorithm/minmax_element.hpp>
 using boost::minmax_element;

#include <iostream> // for debugging.
  using std::cout;
  using std::endl;
  using std::boolalpha;

#include <limits>
  using std::numeric_limits;

#include <vector>
  using std::vector;
#include <set>
  using std::multiset;

#include <utility>
  using std::pair;

#include <boost/svg_plot/detail/auto_axes.hpp>
  using boost::svg::show; // A single STL container.
  using boost::svg::show_all; // Multiple STL containers.
  using boost::svg::range; // Find min and max of a STL container.
  using boost::svg::range_all;// Find min and max of multipler STL containers.
```

This example uses containers to demonstrate autoscaling. Autoscaling must inspect the container in order to find axis ranges that will be suitable. First we create a container and fill with some fictional data.

```
vector<double> my_data;
// Initialize my_data with some entirely fictional data.
my_data.push_back(0.2);
my_data.push_back(1.1); // [1]
my_data.push_back(4.2); // [2]
my_data.push_back(3.3); // [3]
my_data.push_back(5.4); // [4]
my_data.push_back(6.5); // [5]
```

Also included is an 'at limit' value that could confuse autoscaling. Obviously, we do **not** want the plot range to include infinity.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
my_data.push_back(numeric_limits<double>::infinity()); // [6]

try
{ // Ensure error, warning and information messages are displayed by the catch block.
  double mn;
  double mx;
  int good = mnmx(my_data.begin(), my_data.end(), &mn, &mx);
  cout << good << " good values, " << my_data.size() - good << " limit values." << endl;

  using boost::svg::detail::operator<<; // For displaying std::pair.
  svg_1d_plot my_1d_plot; // Construct a plot with all the default constructor values.
  my_1d_plot.x_autoscale(my_data);  // Compute autoscale values for the plot.
  my_1d_plot.limit_color(blue).limit_fill_color(green); // Add limit value styling.

  cout << "my_1d_plot.limit_color() " << my_1d_plot.limit_color() << endl;
  cout << "my_1d_plot.limit_fill_color() " << my_1d_plot.limit_fill_color() << endl;

  my_1d_plot.plot(my_data, "Default 1D"); // Add the one data series, and give it a title.
  my_1d_plot.write("auto_1d_plot_1.svg"); // Write the plot to file.
```

It may be useful to display that range chosen by autoscaling.

```
  cout << "x_range() " << my_1d_plot.x_range() << endl; // x_range()
}
catch(const std::exception& e)
{ // Error, warning and information messages are displayed by the catch block.
  std::cout <<
    "\n""Message from thrown exception was:\n   " << e.what() << std::endl;
}
```

Other STL containers can also be used, for example set, or multiset (allowing duplicates):

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
try
{ // Ensure error, warning and information messages are displayed by the catch block.

multiset<double> my_set;
// Initialize my_set with some entirely fictional data.
my_set.insert(1.2);
my_set.insert(2.3);
my_set.insert(3.4);
my_set.insert(4.5);
my_set.insert(5.6);
my_set.insert(6.7);
my_set.insert(7.8);
my_set.insert(8.9);
// Some examples of showing the set using various iterators:
multiset<double>::const_iterator si;

show(my_data); // Entire container contents,
show(my_set); // for two different types of container.

// show(&my_data[0], &my_data[my_data.size()]); // pointers - wrong! uses more all data ;-)
show(&my_data[0], &my_data[my_data.size()-1]); // pointers, all data.
show(&my_data[1], &my_data[5]); // pointers, part data.
show(my_data.begin(), my_data.end()); // iterators.
show(++(my_data.begin()), --(my_data.end())); // Just the 4 middle values.

vector<double>::const_iterator idb = my_data.begin();
vector<double>::const_iterator ide = my_data.end();
show(idb, ide); // All
++idb; // Move to 2nd value.
--ide; // move back from last value.
show(idb, ide); // Just the 4 middle values.

typedef vector<double>::const_iterator vector_iterator;
pair<vector_iterator, vector_iterator> result = boost::minmax_element(my_data.be↵
gin(), my_data.end());
cout << "The smallest element is " << *(result.first) << endl; // 0.2
cout << "The largest element is  " << *(result.second) << endl; // 6.5
```

Autoscaling can also use two double min and max values provided by the user program. Using `x_autoscale(my_set)` effectively uses the first and last items in the STL container. If the container is sorted, then these are the minimum and maximum values.

```
double min_value = *(my_data.begin());
double max_value = *(--my_data.end());
cout << "my_set min " << min_value << ", max = " << max_value << endl;
```

Function scale_axis is used by autoscale, but is also available for use direct by the user. It accepts parameters controlling the scaling and updates 4 items. Its signature is:

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
void scale_axis(double min_value, double max_value, / Input range
            double* axis_min_value,  double* axis_max_value, double* axis_tick_increment, int* ↵
auto_ticks, / All 4 updated.
              bool check_limits, / Whether to check all values for infinity, NaN etc.
              bool origin, / If true, ensures that zero is a tick value.
            double tight, / Allows user to avoid a small fraction over a tick using another tick.
              int min_ticks, / Minimum number of ticks.
              int steps); // Round up and down to 2, 4, 6, 8, 10, or 5, 10 or 2, 5, 10 systems.
```

```
double axis_min_value; // Values to be updated by autoscale.
double axis_max_value;
double axis_tick_increment;
int axis_ticks;
```

min_value and max_value` could be provided by the user program: usually these values are derived in some way from the user data. Several examples follow.

```
// Scaling using first and last values in container,
// assuming the data are ordered in ascending value,
// for example, set, map, multimap, or a sorted vector or array.
//scale_axis(*my_data.begin(),*(--my_data.end()),
scale_axis(1., 9.,
  &axis_min_value, &axis_max_value, &axis_tick_increment, &axis_ticks,
  false, tol100eps, 6, 0); // Display range.
cout << "scaled min " << axis_min_value << ", max = " << axis_max_value
  << ", increment " << axis_tick_increment << ", axis ticks " << axis_ticks << endl;

// Scaling using two begin & end iterators into STL container,
// scale_axis does finding min and max.
scale_axis(my_data.begin(), my_data.end(),
  &axis_min_value, &axis_max_value, &axis_tick_increment, &axis_ticks,
  true, 3., false, tol100eps, 6); // Display range.
cout << "scaled min " << axis_min_value << ", max = " << axis_max_value
  << ", increment " << axis_tick_increment << ", axis ticks " << axis_ticks << endl;

// Scaling using two begin & end iterators into STL container,
// scale_axis does finding min and max.
scale_axis(my_data[1], my_data[4], // Only middle part of the container used, ignoring 1st and ↵
last values.
  &axis_min_value, &axis_max_value, &axis_tick_increment, &axis_ticks,
  false, tol100eps, 6); // Display range.
cout << "scaled min " << axis_min_value << ", max = " << axis_max_value
  << ", increment " << axis_tick_increment << ", axis ticks " << axis_ticks << endl;

// Scaling using whole  STL vector container,
// scale_axis does finding min and max.
scale_axis(my_data, &axis_min_value, &axis_max_value, &axis_tick_increment, &axis_ticks,
  true, 3., false, tol100eps, 6); // Display range.
cout << "scaled min " << axis_min_value << ", max = " << axis_max_value
  << ", increment " << axis_tick_increment << ", axis ticks " << axis_ticks << endl;

// Scaling using whole STL set container,
// scale_axis does finding min and max.
scale_axis(my_set, &axis_min_value, &axis_max_value, &axis_tick_increment, &axis_ticks,
  true, 3., false, tol100eps, 6); // Display range.
cout << "scaled min " << axis_min_value << ", max = " << axis_max_value
  << ", increment " << axis_tick_increment << ", axis ticks " << axis_ticks << endl;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

However autoscaling may go wrong if the data could contain values that are outside normal limits. Infinity (+ and -), and maximum value, and NaN (Not A Number), are separated by the plot program to allow them to be shown, but separate from 'normal' values. These values similarly can distort automatic scaling: a single infinity would result in useless scaling! When the plot range is set, the maximum and minimum values are checked, and if not normal then an exception will be thrown, and no plot will be produced. However, when autoscaling, it is more useful to ignore 'limit' values. But this means checking all values individually. If it known that all values are normal, for example because they come from some measuring equipment that is known only to produce normal values, it will be much quicker to use 'std::min_max_element' which can take advantange of knowledge of the container.

The function 'autoscale_check_limits(bool)' is provided to control this. If set true, all values will be checked, and those at 'limits' will be ignored in autoscaling. The default is true, to check all values.

If we had many very known normal values to plot and want to autoscale, we might instead opt to ignore these checks, and write:

```
svg_1d_plot my_1d_plot; // Construct a plot with all the default constructor values.

//my_1d_plot.autoscale_check_limits(false);
// This *will throw an exception* if checks are avoided and any values are at 'limits'.

// One could also intercept and change any values calculated by scale_axis here.
// Set the plot to use range and interval from the scale_axis values.
my_1d_plot.x_range(axis_min_value, axis_max_value)
  .x_major_interval(axis_tick_increment);

my_1d_plot.x_autoscale(false);  // Ensure autoscale values are *not* recalculated for the plot.
```

There are also some parameters which can fine-tune the autoscaling to produce more aesthetically pleasing ranges. One can: * enforce the inclusion of zero on the plot. * Specify a minimum number of major ticks. * Specify the steps between major ticks, default 0, or 2 for 2, 4, 6, 8, 10, 5 for 1, 5, 10, or 10 (2, 5, 10). * Avoid values that are a tiny amount over the minimum or maximum from causing an apparently unnecessary tick at the minimum or maximum.

```
// Set some autoscaling parameters:
my_1d_plot.x_with_zero(false);
my_1d_plot.x_min_ticks(10);
my_1d_plot.x_steps(0);
my_1d_plot.x_tight(0.001);

// Show the flags just set.
cout << (my_1d_plot.x_with_zero() ? "x_with_zero, " : "not x_with_zero, ")
  << my_1d_plot.x_min_ticks() << " x_min_ticks, "
  << my_1d_plot.x_steps() << " x_steps, "
  << my_1d_plot.x_tight() << " tightness." << endl;
```

Finally here are some examples of using autoscaling using all or part of containers.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
my_1d_plot.x_autoscale(my_data);  // Use all my_data to autoscale.
cout << "Autoscaled " // Show the results of autoscale:
  "min " << my_1d_plot.x_auto_min_value()
  << ", max "<< my_1d_plot.x_auto_max_value()
  << ", interval " << my_1d_plot.x_auto_tick_interval() << endl; // Autoscaled min 0, max 6.5, ↵
interval 0.5

my_1d_plot.x_autoscale(my_data.begin(), my_data.end());  // Use all my_data to autoscale.

cout << "Autoscaled " // Show the results of autoscale:
  "min " << my_1d_plot.x_auto_min_value()
  << ", max "<< my_1d_plot.x_auto_max_value()
  << ", interval " << my_1d_plot.x_auto_tick_interval() << endl; // Autoscaled min 0, max 6.5, ↵
interval 0.5

my_1d_plot.x_autoscale(my_data[1], my_data[4]);  // Use only part of my_data to autoscale.

cout << "Autoscaled " // Show the results of autoscale:
  "min " << my_1d_plot.x_auto_min_value()
  << ", max "<< my_1d_plot.x_auto_max_value()
  << ", interval " << my_1d_plot.x_auto_tick_interval() << endl; // Autoscaled min 1, max 5.5, ↵
interval 0.5

//my_1d_plot.x_autoscale(true);  // Ensure autoscale values are used for the plot.
// This is also automatically set true by any call of `x_autoscale(some_data);
```

The actual addition of data values to the plot is, of course, quite separate from any autoscaling.

```
my_1d_plot.plot(my_data, "Auto 1D"); // Add the one data series.
cout << "Autoscaled " // Show the results of autoscale:
  " min " << my_1d_plot.x_auto_min_value()
  << ", max "<< my_1d_plot.x_auto_max_value()
  << ", interval " << my_1d_plot.x_auto_tick_interval() << endl; // Autoscaled min 1, max 5.5, ↵
interval 0.5

my_1d_plot.plot(my_set.begin(), my_set.end(), "Auto 1D"); // Add another data series from my_set.
my_1d_plot.plot(my_set, "Auto 1D"); // Add another whole data series from my_set.
my_1d_plot.plot(&my_data[1], &my_data[4], "Auto 1D"); // Add part (1,2 3 but *not* 4) of the ↵
one data series.
//my_1d_plot.plot(&my_set[1], &my_set[4], "Auto 1D"); // operator[] is not defined for set con↵
tainer!

my_1d_plot.write("auto_1d_plot_2.svg"); // Write the plot to file.

using boost::svg::detail::operator<<;
cout << "x_range() " << my_1d_plot.x_range() << endl; // x_range() 1, 5.5
show_1d_plot_settings(my_1d_plot);
}
catch(const std::exception& e)
{ // Error, warning and information messages are displayed by the catch block.
  std::cout <<
    "\n""Message from thrown exception was:\n   " << e.what() << std::endl;
}
```

If necessary, one can obtain a complete listing of all the settings used. This is often useful when the plot does not meet expectations.

The output is:

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
------ Rebuild All started: Project: auto_1d_plot, Configuration: Debug Win32 ------
Deleting intermediate and output files for project 'auto_1d_plot', configuration 'Debug|Win32'
Compiling...
auto_1d_plot.cpp
Linking...
Embedding manifest...
Autorun "j:\Cpp\SVG\debug\auto_1d_plot.exe"
limit value: 1.#INF
6 good values, 1 limit values.
limit value: 1.#INF
x_range() 0, 7
7 values in container: 0.2 1.1 4.2 3.3 5.4 6.5 1.#INF
8 values in container: 1.2 2.3 3.4 4.5 5.6 6.7 7.8 8.9
0.2 1.1 4.2 3.3 5.4 6.5 : 6 values used.
1.1 4.2 3.3 5.4 : 4 values used.
0.2 1.1 4.2 3.3 5.4 6.5 1.#INF : 7 values used.
1.1 4.2 3.3 5.4 6.5 : 5 values used.
0.2 1.1 4.2 3.3 5.4 6.5 1.#INF : 7 values used.
1.1 4.2 3.3 5.4 6.5 : 5 values used.
The smallest element is 0.2
The largest element is  1.#INF
my_set min 0.2, max = 1.#INF
scaled min 1, max = 9, increment 1, axis ticks 9
limit value: 1.#INF
scaled min 0, max = 7, increment 1, axis ticks 8
scaled min 1, max = 6, increment 1, axis ticks 6
limit value: 1.#INF
scaled min 0, max = 7, increment 1, axis ticks 8
scaled min 1, max = 9, increment 1, axis ticks 9
not x_with_zero, 10 x_min_ticks, 0 x_steps, 0.001 tightness.
limit value: 1.#INF
Autoscaled min 0, max 6.5, interval 0.5
limit value: 1.#INF
Autoscaled min 0, max 6.5, interval 0.5
Autoscaled min 1, max 5.5, interval 0.5
Autoscaled  min 1, max 5.5, interval 0.5
x_range() 1, 5.5

axes_on false
background_border_width 1
background_border_color RGB(255,255,0)
background_color RGB(255,255,255)
image_border_margin() 10
image_border_width() 1
coord_precision 3
copyright_date
copyright_holder
description
document_title
image_x_size 500
image_y_size 200
legend_on false
legend_place 2
legend_top_left -1, -1, legend_bottom_right -1, -1
legend_background_color blank
legend_border_color RGB(255,255,0)
legend_color blank
legend_title
legend_title_font_size 14
legend_width 0
legend_lines true
limit points stroke color RGB(119,136,153)
limit points fill color RGB(250,235,215)
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
license_on false
license_reproduction permits
license_distribution permits
license_attribution requires
license_commercialuse permits
plot_background_color RGB(255,255,255)
plot_border_color RGB(255,255,0)
plot_border_width 1
plot_window_on true
plot_window_x 11, 489
plot_window_x_left 11
plot_window_x_right 489
plot_window_y 56, 139
plot_window_y_top 56
plot_window_y_bottom 139.2
title_on true
title ""
title_color blank
title_font_alignment 2
title_font_decoration
title_font_family Verdana
title_font_rotation 0
title_font_size 18
title_font_stretch
title_font_style
x_value_precision 3
x_value_ioflags 200 IOS format flags (0x200) dec.
x_max 5.5
x_min 1
x_axis_on true
x_axis_color() RGB(0,0,0)
x_axis_label_color RGB(0,0,0)
x_axis_value_color RGB(0,0,0)
x_axis_width 1
x_label_on true
x_label
x_label_color blank
x_label_font_family Verdana
x_label_font_size 14
x_label_units
x_label_units_on false
x_major_labels_side left
x_values_font_size 10
x_values_color blank
x_values_precision 3
x_values_ioflags 512
x_major_label_rotation 0
x_major_grid_color RGB(200,220,255)
x_major_grid_on false
x_major_grid_width 1
x_major_interval 0.5
x_major_tick 0.5
x_major_tick_color RGB(0,0,0)
x_major_tick_length 5
x_major_tick_width 2
x_minor_interval 0
x_minor_tick_color RGB(0,0,0)
x_minor_tick_length 2
x_minor_tick_width 1
x_minor_grid_on false
x_minor_grid_color RGB(200,220,255)
x_minor_grid_width 0.5
x_range() 1, 5.5
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
x_num_minor_ticks 4
x_ticks_down_on true
x_ticks_up_on false
x_ticks_on_window_or_axis bottom
x_axis_position x_axis_position intersects Y axis (Y range includes zero)
x_autoscale true
x_autoscale_check_limits true
data lines width 0
```

See auto_1d_plot.cpp for full source code.

1-D Autoscaling Various Containers Examples

First some includes to use Boost.Plot (and some others only needed for this example).

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
#include <boost/svg_plot/svg_1d_plot.hpp>
  using namespace boost::svg;
  using boost::svg::svg_1d_plot;

#include <boost/svg_plot/show_1d_settings.hpp>
// Only needed for showing which settings in use.
 void boost::svg::show_1d_plot_settings(svg_1d_plot&);
 // (Also provides operator<< for std::pair).

#include <boost/algorithm/minmax.hpp>
 using boost::minmax;
#include <boost/algorithm/minmax_element.hpp>
 using boost::minmax_element;

#include <iostream> // for debugging.
  using std::cout;
  using std::endl;
  using std::boolalpha;

#include <limits>
  using std::numeric_limits;

#include <vector>
  using std::vector;
#include <set>
  using std::multiset;

#include <utility>
  using std::pair;

#include <boost/svg_plot/detail/auto_axes.hpp>
  using boost::svg::show; // A single STL container.
  using boost::svg::show_all; // Multiple STL containers.
  using boost::svg::range; // Find min and max of a STL container.
  using boost::svg::range_all;// Find min and max of multipler STL containers.
```

This example uses a few types of containers to demonstrate autoscaling. Autoscaling can inspect the container in order to find axis ranges that will be suitable. First we create a container and fill with some fictional data, and display the values.

```
vector<double> my_data;
// Initialize my_data with some entirely fictional data.
my_data.push_back(0.2);
my_data.push_back(1.1); // [1]
my_data.push_back(4.2); // [2]
my_data.push_back(3.3); // [3]
my_data.push_back(5.4); // [4]
my_data.push_back(6.5); // [5]
show(my_data); // Show entire container contents,
// 6 values in container: 0.2 1.1 4.2 3.3 5.4 6.5
```

Construct a plot , and add some data to the plot.

```
svg_1d_plot my_1d_plot; // Construct a plot with all the default constructor values.
my_1d_plot.title("Demo 1D autoscaling").x_label("X values"); // Add a title and X axis label.
my_1d_plot.plot(my_data, "Auto 1D my_data"); // Add whole data series from my_data.
```

Use x_autoscale to scale the axis, in this most common and simplest case, using all the values.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
my_1d_plot.x_autoscale(my_data);
```

and finally write the SVG to a file.

```
my_1d_plot.write("demo_1d_autoscaling_1.svg"); // Write the plot to file.
```

In a second example, we use a different type of container, a set, and use autoscale in a more advanced way.

```
multiset<double> my_set;
// Initialize my_set with some entirely fictional data.
my_set.insert(1.2);
my_set.insert(2.3);
my_set.insert(3.4);
my_set.insert(4.5);
my_set.insert(5.6);
my_set.insert(6.7);
my_set.insert(7.8);
my_set.insert(8.9);
// Show the set.
multiset<double>::const_iterator si;
show(my_set); // for two different types of container.
// 8 values in container: 1.2 2.3 3.4 4.5 5.6 6.7 7.8 8.9

svg_1d_plot my_1d_plot; // Construct a plot with all the default constructor values.
```

and also override the default controls of scale_axis function used by autoscaling.

```
// Set some autoscaling parameters:
my_1d_plot.x_with_zero(true); // Always include zero in the axis range, even if the data values ↵
don't.
my_1d_plot.x_min_ticks(10); // Ensure more than the default minimum number of ticks.
my_1d_plot.x_steps(0); // Permits axis ticks in steps of 1,2,4,6,8 or 1, 5, 10, or 1, 2, 5, 10.
my_1d_plot.x_tight(0.01); // Prevent values that are only 1% outside the last tick requiring an↵
other tick.

// Show the flags just set.
cout << (my_1d_plot.x_with_zero() ? "x_with_zero, " : "not x_with_zero, ") // x_with_zero
  << my_1d_plot.x_min_ticks() << " x_min_ticks, " // 10
  << my_1d_plot.x_steps() << " x_steps, "  // 0
  << my_1d_plot.x_tight() << " tightness." << endl; // 0.001
```

Purely to show the possibilities with autoscale, we don't use the whole container, but use the two-iterators version of autoscale to **not use the first nor the last values** for autoscaling. (Remember values in set are sorted).

```
my_1d_plot.x_autoscale(++my_set.begin(),--my_set.end());
```

This also sets autoscale(true), but note that x_range() is still not updated. If we want, we can display the ranges chosen by autoscale:

```
cout << " x_auto_min_value " << my_1d_plot.x_auto_min_value()
  << ", x_auto_max _value " << my_1d_plot.x_auto_max_value()
  << ", x_auto_tick_interval " << my_1d_plot.x_auto_tick_interval() << endl;
```

As before, we add the data set to the plot, and write to SVG XML to file. If you inspect the plot, you will see that the lowest data value 1.2 and highest data value 8.9 are no longer shown, because it is now outside the plot window. This is as if we had only written part of the data series.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
  my_1d_plot.plot(++my_set.begin(),--my_set.end(), "Auto 1D my_set"); // Add 'top and tailed' ↵
data series from my_set.
//my_1d_plot.plot(my_set, "Auto 1D my_set"); // Add whole data series from my_set.
 my_1d_plot.write("demo_1d_autoscaling_2.svg"); // Write the plot to file.
```

If we want, we can check the autoscale range used, noting that zero **is** included because we demanded it.

```
cout << "x_range() " << my_1d_plot.x_range() << endl; // x_range() 0, 8
//show_1d_plot_settings(my_1d_plot); // If required.
}
catch(const std::exception& e)
{
```

Try & catch blocks are very useful to display any plot error messages. Otherwise any exceptions thrown will just terminate - silently and unhelpfully.

```
  cout <<"\n""Message from thrown exception was:\n   " << e.what() << endl;
}
} // // Autoscaling with my_set
```

The output is:

```
demo_1d_autoscaling.cpp
Linking...
Embedding manifest...
Autorun "j:\Cpp\SVG\debug\demo_1d_autoscaling.exe"
6 values in container: 0.2 1.1 4.2 3.3 5.4 6.5
8 values in container: 1.2 2.3 3.4 4.5 5.6 6.7 7.8 8.9
x_with_zero, 10 x_min_ticks, 0 x_steps, 0.01 tightness.
 x_auto_min_value 0, x_auto_max _value 8, x_auto_tick_interval 0.5
x_range() 0, 8
Build Time 0:04
```

and the plot:

See demo_1d_autoscaling.cpp for full source code.

1-D Data Values Examples

Showing 1D Data Values Examples

Showing the 1D values of items from the data set. Some of the many possible formatting options are demonstrated.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
As ever, we need a few includes to use Boost.Plot
```

```
#include <boost/svg_plot/svg_1d_plot.hpp>
  using namespace boost::svg;
  using boost::svg::svg_1d_plot;

  #include <boost/svg_plot/show_1d_settings.hpp>
// using boost::svg::show_1d_plot_settings - Only needed for showing which settings in use.

#include <iostream>
  using std::cout;
  using std::endl;
  using std::hex;

#include <vector>
  using std::vector;
```

Some fictional data is pushed into an STL container, here vector<double>:

```
vector<double> my_data;
my_data.push_back(-1.6);
my_data.push_back(2.0);
my_data.push_back(4.2563);
my_data.push_back(0.00333974);
my_data.push_back(5.4);
my_data.push_back(6.556);

try
{ // try'n'catch blocks are needed to ensure error messages from any exceptions are shown.
  svg_1d_plot my_1d_plot; // Construct a plot with all the default constructor values.

  my_1d_plot.title("Default 1D Values Demo") // Add a string title of the plot.
    .x_range(-5, 10) // Add a range for the X-axis.
    .x_label("length (m)"); // Add a label for the X-axis.
```

Add the one data series, `my_data` and a description, and how the data points are to marked, here a circle with a diameter of 5 pixels.

```
my_1d_plot.plot(my_data, "1D Values").shape(round).size(5);
```

To put a value label against each data point, switch on the option:

```
my_1d_plot.x_values_on(true); // Add a label for the X-axis.
```

If the default size and color are not to your taste, set more options, like:

```
my_1d_plot.size(500, 350) // Change from default size
  .x_values_font_size(14) // Change font size for the X-axis value labels.
  .x_values_font_family("Times New Roman") // Change font for the X-axis value labels.
  .x_values_color(red); // Change color from default black to red.
```

The format of the values may also not be ideal, so we can use the normal `iostream precision` and `ioflags` to change, here to reduce the number of digits used from default precision 6 down to a more readable 2, reducing the risk of collisions between adjacent values. (Obviously the most suitable precision depends on the range of the data points. If values are very close to each other, a higher precision wil be needed to differentiate them).

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
my_1d_plot.x_values_precision(2); // precision label for the X-axis value label.
```

We can also prescribe the use of scientific format and force a positive sign:

```
my_1d_plot.x_values_ioflags(std::ios::scientific | std::ios::showpos);
```

By default, any unnecessary spacing-wasting zeros in the exponent field are removed. (If, perversely, the full 1.123456e+012 format is required, the stripping can be switched off with: `my_1d_plot.x_labels_strip_e0s(false);`)

In general, sticking to the defaults usually produces the neatest presentation of the values.

The default value label is horizontal, centered above the data point marker, but, depending on the type and density of data points, and the length of the values (controlled in turn by the `precision` and `ioflags` in use), it is often clearer to use a different orientation. This can be controlled in steps of 45 degrees, using an 'enum rotate_style`.

- `uphill` - writing up at 45 degree slope is often a good choice,
- `upward` - writing vertically up and
- `backup` are also useful.

(For 1-D plots other directions are less attractive, placing the values below the horizontal Y-axis line, but for 2-D plots all writing orientations can be useful).

```
my_1d_plot.x_values_rotation(steepup); // Orientation for the X-axis value labels.

my_1d_plot.x_decor("[ x = ", "", " sec]"); // Note the need for a Unicode space A0.
```

To use all these settings, finally write the plot to file.

```
my_1d_plot.write("demo_1d_values.svg");
```

If chosen settings do not have the effect that you expect, it may be helpful to display them!

(All the myriad settings can be displayed with `show_1d_plot_settings(my_1d_plot)`.)

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
//show_1d_plot_settings(my_1d_plot);

cout << "my_1d_plot.image_size() " << my_1d_plot.image_size() << endl;
cout << "my_1d_plot.image_x_size() " << my_1d_plot.image_x_size() << endl;
cout << "my_1d_plot.image_y_size() " << my_1d_plot.image_y_size() << endl;
cout << "my_1d_plot.x_values_font_size() " << my_1d_plot.x_values_font_size() << endl;
cout << "my_1d_plot.x_values_font_family() " << my_1d_plot.x_values_font_family() << endl;
cout << "my_1d_plot.x_values_color() " << my_1d_plot.x_values_color() << endl;
cout << "my_1d_plot.x_values_precision() " << my_1d_plot.x_values_precision() << endl;
cout << "my_1d_plot.x_values_ioflags() " << hex << my_1d_plot.x_values_ioflags() << endl;
```

```
Output:

demo_1d_values.cpp
Linking...
Embedding manifest...
Autorun "j:\Cpp\SVG\debug\demo_1d_values.exe"
 my_1d_plot.x_values_font_size() 14
my_1d_plot.x_values_font_family() Times New Roman
my_1d_plot.x_values_color() RGB(255,0,0)
my_1d_plot.x_values_precision() 2
 my_1d_plot.x_values_ioflags() 1020
Build Time 0:03
```

Showing 1D Data 'at limit' Values Examples

This example demonstrates showing values that are too small or too large, or NotaNumber.

As ever, we need a few includes to use Boost.Plot

```
#include <boost/svg_plot/svg_1d_plot.hpp>
  using namespace boost::svg;
  using boost::svg::svg_1d_plot;

#include <iostream>
  using std::cout;
  using std::endl;
  using std::hex;

#include <vector>
  using std::vector;

#include <limits>
  using std::numeric_limits;
```

Some fictional data is pushed into an STL container, here vector<double>, including a NaN and + and - infinity:

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
vector<double> my_data;
my_data.push_back(-1.6);
my_data.push_back(2.0);
my_data.push_back(4.2563);
my_data.push_back(-4.0);
my_data.push_back(numeric_limits<double>::infinity());
my_data.push_back(-numeric_limits<double>::infinity());
my_data.push_back(numeric_limits<double>::quiet_NaN());

try
{ // try'n'catch blocks are needed to ensure error messages from any exceptions are shown.
  svg_1d_plot my_1d_plot; // Construct a plot with all the default constructor values.

  my_1d_plot.title("Default 1D NaN and infinities Demo") // Add a string title of the plot.
    .x_range(-5, 5) // Add a range for the X-axis.
    .x_label("length (m)"); // Add a label for the X-axis.
```

Add the one data series, `my_data` and a description, and how the data points are to marked, here a circle with a diameter of 5 pixels.

```
my_1d_plot.plot(my_data, "1D limits").shape(round).size(5);
```

To put a value label against each data point, switch on the option:

```
my_1d_plot.x_values_on(true); // Add a label for the X-axis.
```

To change the default colors (lightgray and whitesmoke) for the 'at limit' point marker to something more conspicuous for this demonstration:

```
my_1d_plot.limit_color(blue);
my_1d_plot.limit_fill_color(pink);
```

To use all these settings, finally write the plot to file.

```
my_1d_plot.write("demo_1d_limits.svg");
```

Note

the +infinity point is marked on the far right of the plot, the -infinity on the far left, but the NaN (Not A Number) is at zero.

To echo the new marker colors chosen:

```
cout << "limit points stroke color " << my_1d_plot.limit_color() << endl;
cout << "limit points fill color " << my_1d_plot.limit_fill_color() << endl;
```

Typical output is:

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
Output:

demo_1d_limits.cpp
Linking...
Embedding manifest...
Autorun "j:\Cpp\SVG\debug\demo_1d_limits.exe"
limit points stroke color RGB(0,0,255)
limit points fill color RGB(255,192,203)
Build Time 0:04
```

See demo_1d_limits.cpp for full source code and sample output.

Real-life Heat flow data

This example shows some real-life data plotted in 1D and as a boxplot.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
// An example to show Heat Flow data from
// http://www.itl.nist.gov/div898/handbook/eda/section4/eda4281.htm
// This data set was collected by Bob Zarr of NIST in January, 1990
// from a heat flow meter calibration and stability analysis.

//#ifdef _MSC_VER
// Path submitted to cure this.
//#  pragma warning (disable: 4510) // boost::array<T,N>' : default constructor could not be gen↵
erated
//#  pragma warning (disable: 4610) // warning C4610: class 'boost::array<T,N>' can never be in↵
stantiated - user defined constructor required
//#endif

#include <boost/svg_plot/svg_1d_plot.hpp>
  using namespace boost::svg;
  using boost::svg::svg_1d_plot;
#include <boost/svg_plot/svg_boxplot.hpp>
  using boost::svg::svg_boxplot;

#include <boost/svg_plot/show_1d_settings.hpp>
 void boost::svg::show_1d_plot_settings(svg_1d_plot&);

#include <iostream> // for debugging.
  using std::cout;
  using std::endl;

#include <vector>
  using std::vector;

#include <limits>
  using std::numeric_limits;

int main()
{
  boost::array<const double, 195> heat_flows = {
  9.206343,
  9.299992,
  9.277895,
  9.305795,
  9.275351,
  9.288729,
  9.287239,
  9.260973,
  9.303111,
  9.275674,
  9.272561,
  9.288454,
  9.255672,
  9.252141,
  9.297670,
  9.266534,
  9.256689,
  9.277542,
  9.248205,
  9.252107,
  9.276345,
  9.278694,
  9.267144,
  9.246132,
  9.238479,
  9.269058,
  9.248239,
  9.257439,
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
9.268481,
9.288454,
9.258452,
9.286130,
9.251479,
9.257405,
9.268343,
9.291302,
9.219460,
9.270386,
9.218808,
9.241185,
9.269989,
9.226585,
9.258556,
9.286184,
9.320067,
9.327973,
9.262963,
9.248181,
9.238644,
9.225073,
9.220878,
9.271318,
9.252072,
9.281186,
9.270624,
9.294771,
9.301821,
9.278849,
9.236680,
9.233988,
9.244687,
9.221601,
9.207325,
9.258776,
9.275708,
9.268955,
9.257269,
9.264979,
9.295500,
9.292883,
9.264188,
9.280731,
9.267336,
9.300566,
9.253089,
9.261376,
9.238409,
9.225073,
9.235526,
9.239510,
9.264487,
9.244242,
9.277542,
9.310506,
9.261594,
9.259791,
9.253089,
9.245735,
9.284058,
9.251122,
9.275385,
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
9.254619,
9.279526,
9.275065,
9.261952,
9.275351,
9.252433,
9.230263,
9.255150,
9.268780,
9.290389,
9.274161,
9.255707,
9.261663,
9.250455,
9.261952,
9.264041,
9.264509,
9.242114,
9.239674,
9.221553,
9.241935,
9.215265,
9.285930,
9.271559,
9.266046,
9.285299,
9.268989,
9.267987,
9.246166,
9.231304,
9.240768,
9.260506,
9.274355,
9.292376,
9.271170,
9.267018,
9.308838,
9.264153,
9.278822,
9.255244,
9.229221,
9.253158,
9.256292,
9.262602,
9.219793,
9.258452,
9.267987,
9.267987,
9.248903,
9.235153,
9.242933,
9.253453,
9.262671,
9.242536,
9.260803,
9.259825,
9.253123,
9.240803,
9.238712,
9.263676,
9.243002,
9.246826,
9.252107,
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
  9.261663,
  9.247311,
  9.306055,
  9.237646,
  9.248937,
  9.256689,
  9.265777,
  9.299047,
  9.244814,
  9.287205,
  9.300566,
  9.256621,
  9.271318,
  9.275154,
  9.281834,
  9.253158,
  9.269024,
  9.282077,
  9.277507,
  9.284910,
  9.239840,
  9.268344,
  9.247778,
  9.225039,
  9.230750,
  9.270024,
  9.265095,
  9.284308,
  9.280697,
  9.263032,
  9.291851,
  9.252072,
  9.244031,
  9.283269,
  9.196848,
  9.231372,
  9.232963,
  9.234956,
  9.216746,
  9.274107,
  9.273776
};

  try
  {
```

try'n'catch block Very useful to ensure you see any error messages!

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
vector<double> heat_flow_data(heat_flows.begin(), heat_flows.end()); // Copy from Boost.array ↵
to vector.
// This might be useful if data is in a C array.

svg_1d_plot heat_flow_1d_plot; // Construct with all the default constructor values.
heat_flow_1d_plot.document_title("NIST Heat_Flow Data") // This text shows on the browser tab.
  .description("NIST Heat_Flow Data")
  .copyright_date("2008-06-19")
  .copyright_holder("Paul A. Bristow, Bob Zarr, NIST")
  .license("permits", "permits", "requires", "permits", "permits"); // Require notice only.
  //see  http://creativecommons.org/licenses/ for details.
// This will generate an XML comment and an author and rights entries thus:
// <!-- SVG Plot Copyright Paul A. Bristow, Bob Zarr, NIST 2008-06-19 -->
// <dc:author><cc:Agent><dc:title>Paul A. Bristow, Bob Zarr, NIST </dc:title> </cc:Agent> </dc:au↵
thor>
// <dc:rights><cc:Agent><dc:title>Paul A. Bristow, Bob Zarr, NIST</dc:title></cc:Agent></dc:rights>

heat_flow_1d_plot.coord_precision(6); // Some rather precise data real-life,
// so use high precision (even at the expense of slightly longer svg files).

heat_flow_1d_plot
  .title("NIST Heat flow data") // Add title for the plot.
  .x_label("heat flow") // Add a label for the X-axis.
  .x_autoscale(heat_flow_data); // Use autoscale from this data.

heat_flow_1d_plot.plot(heat_flows, "NIST heat flows"); // Add a dataset as an array.
//heat_flow_1d_plot.plot(heat_flow_data, "NIST heat flows"); // Add a dataset.
heat_flow_1d_plot.write("./heat_flow_data.svg");
```

Now we use a boxplot to show the same data, including its quartiles and any outliers.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
  svg_boxplot heat_flow_boxplot; // Construct with all the default constructor values.
  heat_flow_boxplot.y_autoscale(heat_flow_data); // Autoscale using this heat_flow data.
  heat_flow_boxplot.title("NIST Heat_Flow Data"); // Give a title.
  heat_flow_boxplot.plot(heat_flow_data, "NIST heat flows"); // Add a dataset.
  heat_flow_boxplot.write("./heat_flow_data_boxplot.svg");
}
catch(const std::exception& e)
{
  std::cout <<
    "\n""Message from thrown exception was:\n   " << e.what() << std::endl;
}
```

See demo_1d_heat_flow_data.cpp for full source code and sample output.

Demonstration of using 1D data that includes information about its Uncertainty

First we need a few includes to use Boost.Plot:

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
#include <boost/svg_plot/uncertain.hpp>
 // using boost::svg::unc; // Holds value and uncertainty formation.
#include <boost/svg_plot/detail/functors.hpp>
//  using boost::svg::detail::unc_1d_convert;
#include <boost/svg_plot/svg_1d_plot.hpp>
  using namespace boost::svg;

#include <iostream>
  using std::cout;
  using std::endl;
#include <vector>
  using std::vector;
#include <iterator>
  using std::ostream_iterator;
#include <algorithm>
  using std::copy;
```

STL vector is used as the container for our two data series, and values are inserted using push_back. Since this is a 1-D plot the order of data values is not important.

```
vector<unc> A_times;
A_times.push_back(unc(3.1, 0.02F, 8));
A_times.push_back(unc(4.2, 0.01F, 14));

vector<unc> B_times;
B_times.push_back(unc(2.1, 0.001F, 30, 0xFF)); // Value, uncertainty, degrees of freedom and ↵
type known.
// (But use of type is not yet implemented.)
B_times.push_back(unc(5.1, 0.025F, 20)); // Value, uncertainty, and degrees of freedom known -
 the usual case.
B_times.push_back(unc(7.8, 0.0025F)); // Value and uncertainty known, but not degrees of freedom.
B_times.push_back(unc(3.4, 0.03F)); // Value and uncertainty known, but not degrees of freedom.
B_times.push_back(unc(6.9)); // Only value known - no information available about uncertainty.

vector<unc> C_times;
C_times.push_back(unc(0.6, 0.1F, 5));
C_times.push_back(unc(1.4, 0.2F, 11));
```

Echo the values input:

```
copy(A_times.begin(), A_times.end(), ostream_iterator<unc>(std::cout, " "));
cout << endl;
copy(B_times.begin(), B_times.end(), ostream_iterator<unc>(std::cout, " "));
cout << endl;
```

The constructor initializes a new 1D plot, called `my_plot`, and also sets all the very many defaults for axes, width, colors, etc.

```
svg_1d_plot my_plot;
```

A few (member) functions that set are fairly self-explanatory:

- title provides a title at the top for the whole plot,
- `legend_on(true)` will mean that titles of data series and markers will display in the legend box.
- `x_range(-1, 11)` sets the axis limits from -1 to +11 (instead of the default -10 to +10).
- `background_border_color(blue)` sets just one of the very many options.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
  my_plot.autoscale_check_limits(false); // default is true.
  my_plot.autoscale_plusminus(3); // default is 3.


my_plot
  .plot_window_on(true)
  .background_border_color(blue)
  .plot_border_color(yellow)
  .plot_border_width(1)
  //.x_ticks_on_window_or_axis(0) // now the default
  .legend_on(false)
  .title("Times")
  .x_range(0, 10)
  .x_label("times (sec)")
  .x_values_on(true)
  .x_values_precision(2)
  .x_values_rotation(slopeup)
  .x_plusminus_color(blue)
  .x_plusminus_on(true)
  .x_df_on(true)
  .x_df_color(green)
  .x_autoscale(A_times)
  ;
```

Then we add our data series, and add optional data series titles (very helpful if we want them to show on the legend).

The A_times mark data points with a red border circle with a green fill, The B_times use a blue vertical line, while C_times use an ellipse whose width (x radius) is from the uncertainty. All the data points are also labelled with their value, and uncertainty and degrees of freedom if known.

```
my_plot.plot(A_times, "A").shape(round).size(10).stroke_color(red).fill_color(green);
my_plot.plot(B_times, "B").shape(vertical_line).stroke_color(blue);
my_plot.plot(C_times, "C").shape(unc_ellipse).fill_color(lightyellow).stroke_color(magenta);
```

Finally, we can write the SVG to a file of our choice.

```
my_plot.write("./demo_1d_uncertainty.svg");
```

The resulting plot is

See demo_1d_uncertainty.cpp for full source code and sample output.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

2D Tutorial

Simple Code Example

```
#include <boost/svg_plot/svg_2d_plot.hpp>
using namespace boost::svg;

#include <map>
using std::multimap; // A more complicated STL container.

int main()
{
  multimap<double, double> map1;
  multimap<double, double> map2;

  // This is random data used purely for example purposes.
  map1[1.] = 3.2;
  map1[1.] = 5.4;
  map1[7.3] = 9.1;

  map2[3.1] = 6.1;
  map2[5.4] = 7.;

  svg_1d_plot my_plot;

  my_plot.title("Race Times")
         .legend_on(true)
         .x_range(-1, 11)
         .background_border_color(black);

  my_plot.plot(map1, "Series 1", blue); // Data point marker color is blue.
  my_plot.plot(map2, "Series 2", orange);

  my_plot.write("simple_2d.svg");
  return 0;
}
```

Resulting Simple_2D Example Image

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Simple_2D Example Breakdown

Let's examine what this does.

```
svg_2d_plot my_plot;
```

This constructor initializes a new 2D plot. This also sets the very many default values.

We could accept all of these and just plot one series of data with

```
my_plot.plot(map1, "Series 1"); // Data point marker color is default black.
```

but we can also add a few optional details:

```
my_plot.title("Race Times")
       .legend_on(true)
       .x_range(-1, 11)
       .background_border_color(black);
```

All of the setter methods are fairly self-explanatory. To walk through it once,

- The title, which will appear at the top of the graph, will say "Race Times".
- `legend_on(true)` means that the legend box will show up (by default at top right).
- `x_range(-1, 11)` means that the axis displayed will be between -1 and 11, as you can see in the above images.
- `background_border_color(black)` sets the border around the image to `black`. Ordinarily it is left to be the color of the whole image background (and so no border of the plot area will be visible).

```
my_plot.plot(map1, "Series 1", blue);
my_plot.plot(map2, "Series 2", orange);
```

This draws `map1` and `map2` to `my_plot`. As many containers as you want can be drawn to `my_plot`. Eventually the plot will become cluttered and confusing, when creating other plot(s) becomes sensible!

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

The name of the series is `"Race times"`, and that text will show in the legend box. These are the two required parameters for this function call. There are many optional parameters, as seen in the section Getting More Out of the `plot()` Function

Finally

```
my_plot.write("simple_2d.svg");
```

writes plot `my_plot` to the file "simple_2d.svg".

You can view this file with most internet browsers, and other programs too.

Tutorial: Fuller Layout Example

```
#include <boost/svg_plot/svg_2d_plot.hpp>
using namespace boost::svg;
#include <map>
using std::multimap;
#include <cmath> // for sqrt

// Some example functions:
double f(double x)
{
  return sqrt(x);
  // Note: negative values will all return NaN.
}

double g(double x)
{
  return -2 + x * x;
}

double h(double x)
{
  return -1 + 2 * x;
}

int main()
{
  multimap<double, double> data1, data2, data3;

  for(double i = 0; i <= 10.; i += 1.)
  { // Evaluate the functions as demonstration data.
    data1[i] = f(i);
    data2[i] = g(i);
    data3[i] = h(i);
  }

  svg_2d_plot my_plot; // Construct a new plot.

  // Override a few default settings with our choices:

  // Size of SVG image and X and Y range settings.
  my_plot.image_size(700, 500)
         .x_range(-1, 10)
         .y_range(-5, 100)

  // Text settings.
  my_plot.title("Plot of Mathematical Functions")
         .title_font_size(25)
         .x_label("Time in Months");
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
  // Commands.
  my_plot.legend_on(true)
         .plot_window_on(true)
         .x_label_on(true)
         .x_major_labels_on(true);

  // Color settings.
  my_plot.background_color(svg_color(67, 111, 69))
         .legend_background_color(svg_color(207, 202,167))
         .legend_border_color(svg_color(102, 102, 84))
         .plot_background_color(svg_color(136, 188, 126))
         .title_color(white);

  // X axis settings.
  my_plot.x_major_interval(2)
         .x_major_tick_length(14)
         .x_major_tick_width(1)
         .x_minor_tick_length(7)
         .x_minor_tick_width(1)
         .x_num_minor_ticks(3)

  // Y axis settings.
         .y_major_tick(10)
         .y_num_minor_ticks(2);

  // Legend settings.
  my_plot.legend_title_font_size(15);

  // Add the 3 data series to the plot, using different markers and line colors.
  my_plot.plot(data1, "Sqrt(x)").fill_color(red);
  my_plot.plot(data2, "-2 + x^2").fill_color(orange).size(5);
 my_plot.plot(data3, "-1 + 2x").fill_color(yellow).bezier_on(true).line_color(blue).shape(square);
  // Note how the options can be chained.

  my_plot.write("2d_full.svg");

  return 0;
}
```

This produces the following output:

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

A little bit of color customization goes a **very** long way!

Tutorial: 2D Special Features

Y-Axis ticks and tick value label position

```
int main()
{
```

Following previous examples, we set up two containers for two data series.

```
vector<double> dan_times;
vector<double> elaine_times;

dan_times.push_back(3.1);
dan_times.push_back(4.2);
elaine_times.push_back(2.1);
elaine_times.push_back(7.8);

svg_1d_plot my_plot;

// Adding some generic settings.
my_plot.background_border_color(black)
       .legend_on(true)
       .plot_window_on(true)
       .title("Race Times")
       .x_range(-1, 10);
```

We add tastelessly color the grids for both major and minor ticks, and switch both grids on.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
my_plot.x_major_grid_color(pink)
       .x_minor_grid_color(lightgray);

my_plot.x_major_grid_on(true)
       .x_minor_grid_on(true);
```

Also we specify the position of the labelling of the X-axis. It can be controlled using values in the enum x_axis_intersect.

```
enum x_axis_intersect
{ //! \enum x_axis_intersect
  bottom = -1, // On the bottom of the plot window.
  x_intersects_y = 0, // On the Y axis (in the middle of the plot window).
  top = +1 // On the top of the plot window.
};
```

For this example, we choose to show the X axis and tick value labels at the top of the plot window.

```
  my_plot.x_ticks_on_window_or_axis(top); // on top, not on axis.

  // Write to plot.
  my_plot.plot(dan_times, "Dan").stroke_color(blue);
  my_plot.plot(elaine_times, "Elaine").stroke_color(orange);

  // Write to file.
  my_plot.write("./demo_1d_x_external.svg");
  return 0;
} // int main()
```

producing this plot

See demo_1d_x_external.cpp for full source code.

X-Axis Tick value labels precision, iosflags, font family and size

As ever, we need a few includes to use Boost.Plot and an STL container.

Some fictional data is pushed into an STL container, here vector<double>:

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
vector<double> my_data;
my_data.push_back(-1.6);
my_data.push_back(4.2563);
my_data.push_back(0.00333974);
my_data.push_back(5.4);
my_data.push_back(6.556);

try
{ // try'n'catch blocks are needed to ensure error messages from any exceptions are shown.
  svg_1d_plot my_1d_plot; // Construct a plot with all the default constructor values.

  my_1d_plot.title("Demo 1D Tick Values") // Add a string title of the plot.
    .x_range(-5, 10) // Add a range for the X-axis.
    .x_label("temp (&#x00B0;C)"); // Add a label for the X-axis, using Unicode degree symbol.
```

Add the one data series, `my_data` and a description, and how the data points are to be marked, a circle with a diameter of 7 pixels.

```
my_1d_plot.plot(my_data, "1D Values").shape(round).size(7);
```

If the default size and color are not to your taste, set more options, like:

```
my_1d_plot.size(500, 150) // Change plot window from the default image size.
  // Change the X-axis label style:
  .x_axis_label_color(green)
  .x_label_font_family("Arial")
  .x_label_font_size(18)

  // Change the style of the X (major) ticks:
  .x_ticks_values_color(magenta)
  //.x_ticks_values_font_family("Times New Roman")
  .x_ticks_values_font_family("arial")
  .x_ticks_values_font_size(15)
```

The format of the tick value labels may not suit your data and its range, so we can use the normal `iostream precision` and `ioflags` to change, here to reduce the number of digits used from default precision 6 down to a more readable 2, reducing the risk of collisions between adjacent values. If values are very close to each other (a small range on the axis), a higher precision wil be needed to differentiate them). We could also prescribe the use of scientific format and force a positive sign: By default, any unnecessary spacing-wasting zeros in the exponent field are removed.

If, perversely, the full 1.123456e+012 format is required, the stripping can be switched off with: `my_1d_plot.x_ticks_values_strip_e0s(false);`

```
 // Change the format from the default "-4", "-2", "0" "2", "4" ...
 // (which makes a 'best guess' at the format)
 // to "-4.00", "-2.00", ..."+2.00", "4.00"
 // showing trailing zeros and a leading positive sign.
 .x_ticks_values_ioflags(ios_base::fixed | std::ios::showpos)
 .x_ticks_values_precision(1) //

 // One could use ios_base::scientific for e format.
 //.x_ticks_values_ioflags(ios_base::fixed | std::ios::showpos )
 //.x_ticks_values_ioflags(std::ios::showpoint | std::ios::showpos)
 //.x_ticks_values_ioflags(std::ios::scientific)
;
```

To use all these settings, finally write the plot to file.

```
my_1d_plot.write("demo_1d_tick_values.svg");
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

If chosen settings do not have the effect that you expect, it may be helpful to display them.

(All the myriad settings can be displayed with `show_1d_plot_settings(my_1d_plot)`.)

```
//show_1d_plot_settings(my_1d_plot);

cout << "my_1d_plot.image_size() " << my_1d_plot.image_size() << endl;
cout << "my_1d_plot.image_x_size() " << my_1d_plot.image_x_size() << endl;
cout << "my_1d_plot.image_y_size() " << my_1d_plot.image_y_size() << endl;

cout << "my_1d_plot.x_axis_label_color() " << my_1d_plot.x_axis_label_color() << endl;
cout << "my_1d_plot.x_label_font_family() " << my_1d_plot.x_label_font_family() << endl;
cout << "my_1d_plot.x_label_font_size() " << my_1d_plot.x_label_font_size() << endl;

cout << "my_1d_plot.x_ticks_values_font_family() " << my_1d_plot.x_ticks_values_font_fam↵
ily() << endl;
cout << "my_1d_plot.x_ticks_values_font_size() " << my_1d_plot.x_ticks_values_font_size() << endl;
cout << "my_1d_plot.x_ticks_values_color() " << my_1d_plot.x_ticks_values_color() << endl;

cout << "my_1d_plot.x_ticks_values_precision() " << my_1d_plot.x_ticks_values_precision() << endl;
cout << "my_1d_plot.x_ticks_values_ioflags() " << hex << my_1d_plot.x_ticks_values_io↵
flags() << endl;
```

See demo_1d_ticks_values.cpp for full source code.

producing this plot

See demo_1d_tick_values.cpp for full source code.

Y-Axis Grid Lines

If you would like horizontal Y grid lines that go across the graph, you can make the following call to `svg_2d_plot`:

```
my_plot.y_major_grid_on(true)
       .y_minor_grid_on(true);
```

To style it, you could use the following calls:

```
my_plot.y_major_grid_color(lightgray) // Darker color for major grid.
       .y_minor_grid_color(whitesmoke); // Lighter color for minor grid.
```

This will produce the following plot image:

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

The source of this examples is at ..\..\example\2d_y_grid.cpp

Similarly you can have horizontal and/or vertical lines:

```
my_plot.x_major_grid_on(true)
       .x_minor_grid_on(true);
```

and color it similarly for faint blues like old fashioned graph paper:

```
my_plot.y_major_grid_color(svg_color(200, 220, 255)) // Darker color for major grid.
       .y_minor_grid_color(svg_color(240, 240, 255); // lighter color for minor grid.
```

and to make the major grid much wider than the minor grid:

```
my_plot.y_major_grid_width(4)
       .y_minor_grid_width(1)
```

Fill the area between the plot and the axis

When there is a call to the plot() method, define `area_fill_color`

```
multimap<double, double> my_data;
svg_2d_plot my_plot;

my_plot.plot(my_data, "Data", area_fill_color(red));
```

This produces the following images:

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

from source at ..\..\example\2d_area_fill.cpp

Curve Interpolation

If you would like an **interpolated curve** shown over your data (rather than the default straight lines joining points), simply use the following command:

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
my_plot.plot(data, "Series 1", bezier_on(true);
```

This produces something like the following images:

Warning

The `_bezier_curve` feature still displays undesired behavior in extreme circumstances. Do not use this feature with curves that have a numeric_limit value (`-NaN`, for example), or with data that has high irregularity in X-Axis spacing (for example, a clump of points between (0, 1) on the X axis, with the next one at 100 on the X axis.) In general, curves must be reasonably well-behaved to be smoothable. In general, using more data points will make smoothing work better, but will increase svg file sizes.

2-D Data Values Examples

Showing 2d Data Values Examples

As always, we need a few includes to use Boost.Plot

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
#include <boost/svg_plot/svg_2d_plot.hpp>
  using namespace boost::svg;
  using boost::svg::svg_2d_plot;

#include <iostream>
  using std::cout;
  using std::endl;
  using std::dec;
  using std::hex;

#include <map>
  using std::map;
```

Some fictional data is pushed into an STL container, here map: This example uses a single map to demonstrate autoscaling. We create a map to hold our data series.

```
map<const double, double> my_data;
```

Inserting some fictional values also sorts the data. The index value in [] is the x value.

```
//my_data[1.1] = 3.2;
//my_data[7.3] = 9.1;
my_data[2.12] = 2.4394;
my_data[5.47] = 5.3861;

try
{ // try'n'catch blocks are needed to ensure error messages from any exceptions are shown.
  svg_2d_plot my_2d_plot; // Construct a plot with all the default constructor values.

  my_2d_plot.title("Default 2d Values Demo") // Add a string title of the plot.
    .x_range(-5, 10) // Add a range for the X-axis.
    .x_label("length (m)"); // Add a label for the X-axis.
```

Add the one data series, `my_data` and a description, and how the data points are to be marked, here a circle with a diameter of 5 pixels.

```
my_2d_plot.plot(my_data, "2d Values").shape(round).size(5).line_on(false);
```

To put a value label against each data point, switch on the option:

```
//my_2d_plot.x_values_on(true); // Add a label for the X-axis.
//my_2d_plot.y_values_on(true); // Add a label for the X-axis.
my_2d_plot.xy_values_on(true); // Add a label for the X-axis.
```

If the default size and color are not to your taste, set more options, like:

```
my_2d_plot.x_values_font_size(16) // Change font size for the X-axis value labels.
  .x_values_font_family("Times New Roman") // Change font for the X-axis value labels.
  .x_values_color(red); // Change X values color from default black to red.

my_2d_plot.y_values_font_size(14) // Change font size for the Y-axis value labels.
  .y_values_font_family("Arial") // Change font for the Y-axis value labels.
  .y_values_color(blue); // Change Y color from default black to blue.
```

The format of the values may also not be ideal, so we can use the normal `iostream precision` and `ioflags` to change, here to reduce the number of digits used from default precision 6 down to a more readable 2, reducing the risk of collisions between adjacent

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

values. (Obviously the most suitable precision depends on the range of the data points. If values are very close to each other, a higher precision will be needed to differentiate them). For measurement of typical precision, 2 or 3 decimal places will suffice.

```
my_2d_plot.x_values_precision(3); // precision label for the X-axis value label.
my_2d_plot.y_values_precision(5); // precision label for the Y-axis value label.
```

We can also prescribe the use of scientific, fixed format and/or force a positive sign:

```
//my_2d_plot.x_values_ioflags(std::ios::scientific | std::ios::showpos);
//my_2d_plot.x_values_ioflags(std::ios::scientific);
//my_2d_plot.y_values_ioflags(std::ios::fixed);
```

By default, any unnecessary spacing-wasting zeros in the exponent field are removed. Stripping "e+000" may appear to mean that the normal effect of `scientific` is not working. (If, probably perversely, the full 1.123456e+012 format is required, the stripping can be switched off with: `my_2d_plot.x_labels_strip_e0s(false);`)

In general, sticking to the default ioflags usually produces the neatest presentation of values.

```
my_2d_plot.x_plusminus_on(true); // unc label for the X-axis value label.
my_2d_plot.x_df_on(true); // df label for the X-axis value label.

my_2d_plot.y_plusminus_on(true); // unc label for the X-axis value label.
my_2d_plot.y_df_on(true); // df label for the X-axis value label.
```

The default value label is horizontal, centered above the data point marker, but, depending on the type and density of data points, and the length of the values (controlled in turn by the `precision` and `ioflags` in use), it is often clearer to use a different orientation. This can be controlled in steps of 45 degrees, using an 'enum rotate_style`.

- `leftward` - writing level with the data point but to the left.
- `rightward` - writing level with the data point but to the right.
- `uphill` - writing up at 45 degree slope is often a good choice,
- `upward` - writing vertically up and
- `backup` - writing to the left are also useful.

(For 1-D plots other directions are less attractive, placing the values below the horizontal Y-axis line, but for 2-D plots all writing orientations can be useful).

```
 my_2d_plot.x_values_rotation(rightward); // Orientation for the Y-axis value labels.
 //my_2d_plot.x_values_rotation(horizontal); // Orientation for the X-axis value labels.
 //my_2d_plot.x_values_rotation(uphill); // Orientation for the X-axis value labels.
// my_2d_plot.y_values_rotation(downhill); // Orientation for the Y-axis value labels.
 //my_2d_plot.x_values_rotation(leftward); // Orientation for the X-axis value labels.
 //my_2d_plot.y_values_rotation(rightward); // Orientation for the Y-axis value labels.

// my_2d_plot.x_plusminus_on(true); // Show Uncertainty for X-axis value labels.
// my_2d_plot.x_df_on(true); // Show Degrees of freedom (n-1) for X-axis value labels.
 //my_2d_plot.y_plusminus_on(true); // Uncertainty for X-axis value labels.
```

To use all these settings, finally write the plot to file.

```
my_2d_plot.write("demo_2d_values.svg");
```

If chosen settings do not have the expected effect, is may be helpful to display them.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

(All settings can be displayed with `show_2d_plot_settings(my_2d_plot)`.)

```
cout << "my_2d_plot.x_values_font_size() " << my_2d_plot.x_values_font_size() << endl;
cout << "my_2d_plot.x_values_font_family() " << my_2d_plot.x_values_font_family() << endl;
cout << "my_2d_plot.x_values_color() " << my_2d_plot.x_values_color() << endl;
cout << "my_2d_plot.x_values_precision() " << my_2d_plot.x_values_precision() << endl;
cout << "my_2d_plot.x_values_ioflags() " << hex << my_2d_plot.x_values_ioflags() << dec << endl;

cout << "my_2d_plot.y_values_font_size() " << my_2d_plot.y_values_font_size() << endl;
cout << "my_2d_plot.y_values_font_family() " << my_2d_plot.y_values_font_family() << endl;
cout << "my_2d_plot.y_values_color() " << my_2d_plot.y_values_color() << endl;
cout << "my_2d_plot.y_values_precision() " << my_2d_plot.y_values_precision() << endl;
cout << " my_2d_plot.y_values_ioflags() " << hex << my_2d_plot.y_values_ioflags() << dec << endl;
```

```
Output:

demo_2d_values.cpp
Linking...
Embedding manifest...
Autorun "j:\Cpp\SVG\debug\demo_2d_values.exe"
my_2d_plot.x_values_font_size() 16
my_2d_plot.x_values_font_family() Times New Roman
my_2d_plot.x_values_color() RGB(255,0,0)
my_2d_plot.x_values_precision() 3
my_2d_plot.x_values_ioflags() 200
my_2d_plot.y_values_font_size() 14
my_2d_plot.y_values_font_family() Arial
my_2d_plot.y_values_color() RGB(0,0,255)
my_2d_plot.y_values_precision() 5
 my_2d_plot.y_values_ioflags() 200
```

And the plot:

Showing 2d Data 'at limit' Values Examples

Some values are 'at limit' - infinite, NotANimber (NaN) or near the maximum possible for the floating-point type. These values are displayed differently (and not used for autoscaling).

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

An example to demonstrate plotting 2D 'at limts' values including NaN and + and - infinity.

As ever, we need the usuall includes to use Boost.Plot.

Some fictional data is pushed into an STL container, here map:

```
map<double, double> my_data;
```

Inserting some fictional values also sorts the data. The map index value in [] is the x value, so mydata[x] = y.

First some normal valued points, not 'at limits'.

```
my_data[1.1] = 3.2;
my_data[4.3] = 3.1;
my_data[0.25] = 1.4;
```

Now some values including + and - infinity:

```
my_data[3] = numeric_limits<double>::quiet_NaN(); // marker at x = 3, y = 0
my_data[0.] = numeric_limits<double>::quiet_NaN(); // Marker at 0,0
my_data[1.] = numeric_limits<double>::infinity(); // Marker at 1, top
my_data[-1] = -numeric_limits<double>::infinity(); // Marker at -1, bottom
my_data[+numeric_limits<double>::infinity()] = +1.; // Marker at right, 1
my_data[-numeric_limits<double>::infinity()] = -1.; // Marker at left, -1
my_data[+numeric_limits<double>::max()] = +2.;  // Marker at right, 2
my_data[-numeric_limits<double>::max()] = +2.; // Marker at left, 2
my_data[-numeric_limits<double>::max() /2] = +3.; // Value near to max, marker left, 3
my_data[numeric_limits<double>::infinity()] = numeric_limits<double>::infinity(); // Top right.
my_data[-numeric_limits<double>::infinity()] = -numeric_limits<double>::infinity(); // Bottom left.
```

Caution

Using map (rather than multimap that allows duplicates) some assignments values overwrite, and so not all display as they do individually. In particular, an X value of quiet_NaN() causes a overwrite of the lowerest value (because NaNs never compare equal). So avoid NaN as an X value.

```
try
{ // try'n'catch blocks are needed to ensure error messages from any exceptions are shown.
  svg_2d_plot my_2d_plot; // Construct a plot with all the default constructor values.

  my_2d_plot.title("Default 2D 'at limits' NaN and infinities Demo") // Add a string title of ↵
the plot.
    .x_range(-5, 5) // Add a range for the X-axis.
    .y_range(-5, 5) // Add a range for the Y-axis
    .x_label("time (s)"); // Add a label for the X-axis.
```

Add the one data series, `my_data` and a description, and how the data points are to be marked, here a circle with a diameter of 5 pixels.

```
svg_2d_plot_series& my_series = my_2d_plot.plot(my_data, "2D limits").shape(round).size(5);
```

We can also keep note of the plot series and use this to interrogate how many normal and how many 'at limit' values.

```
cout << my_series.values_count() << " normal data values in series." << endl;
cout << my_series.limits_count() << " 'at limits' data values in series."<< endl;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

To put a value label against each data point, switch on the option:

```
//  my_2d_plot.x_values_on(true).y_values_on(true).x_values_font_size(12).y_values_font_size(12); ↵
// Add the X-axis and Y-axis values.
// This displays x horizontally and Y downward.
my_2d_plot.xy_values_on(true).x_values_font_size(12).y_values_font_size(12); // Add the X-axis ↵
and Y-axis values.
// This displays X above and Y below.
```

To change the default colors (lightgray and whitesmoke) for the 'at limit' point marker to something more conspicuous for this demonstration:

```
my_2d_plot.limit_color(blue);
my_2d_plot.limit_fill_color(pink);
```

To use all these settings, finally write the plot to file.

```
my_2d_plot.write("demo_2d_limits.svg");
```

```
Note the +infinity point is marked on the far right of the plot, the -infinity on the far ↵
left, but the NaN (Not A Number is at zero).
```

To echo the new marker colors chosen:

```
cout << "limit points stroke color " << my_2d_plot.limit_color() << endl;
cout << "limit points fill color " << my_2d_plot.limit_fill_color() << endl;
```

[demo_2d_limits_output]

2-D Autoscaling Examples

Autoscale 2D Examples

First we need a few includes to use Boost.Plot

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
#include <boost/svg_plot/svg_2d_plot.hpp>
  using namespace boost::svg;
#include <utility>
  using std::pair;
#include <map>
  using std::map;
#include <set>
  using std::multiset;
#include <iostream>
  using std::cout;
  using std::endl;

#include <limits>
  using std::numeric_limits;

#include <boost\math\special_functions\fpclassify.hpp>
  using boost::math::isfinite;

  // demo of getting the max and min of x and y
template <class T> // T an STL container: array, vector ...
void s(const T& container, // Container Data series to plot - entire container.
               // (not necessarily ordered, so will find min and max).
               double* x_min,  double* x_max,
               double* y_min,  double* y_max
               )
{
  typedef T::const_iterator iter;
  std::pair<iter, iter> result = boost::minmax_element(container.begin(), container.end());
  // minmax_element is efficient for maps because can use knowledge of being sorted,
  // BUT only if it can be assumed that no values are 'at limits',
  // infinity, NaN, max_value, min_value, denorm_min.
  // Otherwise it is necessary to inspect all values individually.
  pair<const double, double> px = *result.first;
  pair<const double, double> py = *result.second;
  *x_min = px.first;
  *x_max = py.first;
  *y_min = px.second;
  *y_max = py.second;

  cout << "s x_min " << *x_min << ", x_max " << *x_max << endl; // x_min 1, x_max 7.3
  cout << "s y_min " << *y_min << ", y_max " << *y_max << endl; // y_min 3.2, y_max 9.1
} // template <class T> int scale_axis  T an STL container: array, vector ...
```

This example uses a single map to demonstrate autoscaling. We create a map to hold our data series.

```
map<const double, double> my_map;
```

Inserting some fictional values also sorts the data. The index value in [] is the x value.

```
my_map[1.1] = 3.2;
my_map[7.3] = 9.1;
my_map[2.1] = 5.4;
```

Also include some 'at limits' values that might confuse autoscaling.

```
my_map[99.99] = numeric_limits<double>::quiet_NaN();
my_map[999.9] = numeric_limits<double>::infinity();
my_map[999.] = +numeric_limits<double>::infinity();
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Next a 2D plot is created using defaults for the very many possible settings.

```
try
{ // try'n'catch clocks are needed to ensure error messages from any exceptions are shown.
```

Construct `myplot` and add at least a title, specify the both x and y axes are to use autoscaling, and add the one data series to be plotted.

```
svg_2d_plot my_plot;
my_plot.title("Autoscale example"); // Add a title.
my_plot.xy_autoscale(my_map); // Specify that both x and y axes are to use autoscaling,
my_plot.plot(my_map); // Add the one data series to be plotted
my_plot.write("./auto_2d_plot.svg"); // And write the SVG image to a file.
```

We can show the ranges chosen by autoscaling;

```
cout << "X min " << my_plot.x_range().first << ", X max " << my_plot.x_range().second << endl;
cout << "Y min " << my_plot.y_range().first << ", Y max "  << my_plot.y_range().second << endl;
```

Had we know that there were no 'at limits' values, we could have chosen to skip the checks. This might be important for speed if there were thousands of data values.

```
my_plot.autoscale_check_limits(false);  // Skip checks for speed.
```

The possible cost is that it will fail at run-time if there are any infinite or NaNs.

```
We could also chose to autoscale either of the axes separately, for example:
```

```
my_plot.y_autoscale(0.4, 9.3); // autoscale using two doubles.
```

which will chose a neater scale from 0 to 10 for the Y axis.

```
my_plot.write("./auto_2d_plot2.svg"); // And write another SVG image to a file.
```

We can show the ranges chosen by autoscaling;

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
cout << "X min " << my_plot.x_range().first << ", X max " << my_plot.x_range().second << endl;
cout << "Y min " << my_plot.y_range().first << ", Y max "  << my_plot.y_range().second << endl;
```

```
Output:

auto_2d_plot.cpp
Linking...
Embedding manifest...
Autorun "j:\Cpp\SVG\debug\auto_2d_plot.exe"
2 goods, 2.1 5.4
3 goods, 7.3 9.1
limit value: 99.99 1.#QNAN
limit value: 999 1.#INF
limit value: 999.9 1.#INF
x_min 1.1, x_max 7.3
y_min 3.2, y_max 9.1
limits 3
1 8
3 10
Build Time 0:21


Linking...
Embedding manifest...
Autorun "j:\Cpp\SVG\debug\auto_2d_plot.exe"
X min 1, X max 8
Y min 3, Y max 10
Build Time 0:03
```

The plot is

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Demonstration of using 1D data that includes information about its uncertainty

First we need some includes to use Boost.Plot and C++ Standard Library:

```
#include <boost/svg_plot/svg_2d_plot.hpp>
using namespace boost::svg;

#include <boost/svg_plot/show_2d_settings.hpp>
// using boost::svg::show_2d_plot_settings - Only needed for showing which settings in use.
#include <boost/svg_plot/uncertain.hpp>
using boost::svg::unc;
#include <boost/svg_plot/detail/pair.hpp>
// using boost::svg::detail::pair; operator<<

//#include <algorithm>
//#include <functional>

#include <map>
  using std::map;
  using std::multimap;

#include <utility>
  using std::pair;
  using std::make_pair;

#include <vector>
   using std::vector;

#include <cmath>
   using std::sqrt;
   using std::pair;
#include <iostream>
   using std::cout;
   using std::endl;
   using std::scientific;
   using std::hex;
   using std::ios;
   using std::boolalpha;

#include <iterator>
   using std::ostream_iterator;
```

STL map is used as the container for our two data series, and pairs of values and their uncertainty information (approximately standard deviation and degrees of freedom) are inserted using push_back. Since this is a 2-D plot the order of data values is important.

```
unc u1(1.23, 0.56F, 7); // For an X value.
cout << "u1 = " << u1 << endl; // u1 = 1.23+-0.056 (7)
unc u2(3.45, 0.67F, 9); // For a Y value.
pair<unc, unc> mp1 = make_pair(u1, u2); // XY pair of values.
cout << mp1 << endl; // 1.23+-0.056 (7), 2.345+-0.067 (9)

map<unc, unc> data1; // Container for XY points.
data1.insert(mp1); // u1, u2 = 1.23+-0.056 (7), 2.345+-0.067 (9)
data1.insert(make_pair(unc(4.1, 0.4F, 7), unc(3.1, 0.3F, 18))); //
data1.insert(make_pair(unc(-2.234, 0.03F, 7), unc(-8.76, 0.9F, 9)));
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Make very sure you don't forget either unc() like this `data1.insert(make_pair((-2.234, 0.12F, 7),(-8.76, 0.56F, 9)));` because, by the bizarre operation of the comma operator, the result will be an integer! So you will astonished to find that the values will be the pair of \b degrees of freedom, (7, 9) and the other parts of unc will be undefined!/n

```
Echo the values input:
```

```
cout << data1.size() << " XY data pairs:" << endl;
copy(data1.begin(), data1.end(), ostream_iterator<pair<unc, unc> >(cout, "\n"));
cout << endl;

svg_2d_plot my_plot;
```

If you can be confident that the data set(s) only contains normal, valid data, so none are 'at limits' - too big or too small to be meaningful, infinite or NaN (NotANumber), then these checks can be skipped (for speed). An instrument or operator input might be known to provide only normal data. For this example, we know this is true, so override the default autoscale_check_limits(true).

```
//my_plot.autoscale_check_limits(false);
```

The default is autoscale_plusminus(3.) so that confidence ellipses at 1, 2 and 3 (uncertainty nominally standard deviations) are all within the plot window, but if you are less interested in seeing the 2 and 3 ellipses, you could risk the outer edges spilling over the borders by reducing autoscale_plusminus, for example, to 1.5, down to zero.

```
my_plot.autoscale_plusminus(1.5); // default is 3.
```

Use data set `data` to autoscale (you can use a different data set to scale from the one you chose to plot).

```
my_plot.xy_autoscale(data1);
my_plot
  .x_label("times (sec)")
  .xy_values_on(true)
  // Note use of Unicode space in all string - ANSI space has no effect.
  //.x_decor("", ", ") // Keep all on one line using spearator starting with a comma.
  .x_decor("","") // Split onto two lines because X separator does not start with comma.
  .y_decor("   ", "time = ", " sec")
  // Note a few padding spaces used to get Y values to lie more nearly under X values.
  // This is only necessary when label are not horizontal.
  .x_values_rotation(slopeup)
  .x_values_font_size(16)
  .x_plusminus_on(true)
  .x_plusminus_color(cyan)
  .x_df_on(true)
  .x_df_color(magenta)
  .x_values_font_family("Times New Roman")

  .y_label("distance (km)")
  .y_values_rotation(uphill)
  .y_values_font_family("Arial") // Different from X just to show effect.
  .y_plusminus_on(true)
  .y_plusminus_color(red)
  .y_df_on(true)
  .y_df_color(green)
```

The default uncertainty ellipse colors (that apply to both X and Y axes) can be changed thus:

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
   .one_sd_color(lightblue)
   .two_sd_color(svg_color(200, 230, 255))
   .three_sd_color(svg_color(230, 255, 255))
  ;

my_plot.plot(data1, "data1").shape(unc_ellipse);

my_plot.write("./demo_2d_uncertainty");

//show_2d_plot_settings(my_plot);
```

The resulting plot is

See demo_2d_uncertainty.cpp for full source code and sample output.

Demonstration of adding lines and curves, typically a least squares fit

First we need some includes to use Boost.Plot and C++ Standard Library:

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
#include <boost/svg_plot/svg_2d_plot.hpp>
using namespace boost::svg;

#include <boost/svg_plot/show_2d_settings.hpp>
// using boost::svg::show_2d_plot_settings - Only needed for showing which settings in use.

#include <map>
  using std::map;
  using std::multimap;

#include <iostream>
   using std::cout;
   using std::endl;
   using std::boolalpha;

int main()
{
  try
  {
//[demo_2d_lines_2
```

This shows how to add lines to a plot, for example, for example a least-squares 'function' fit line. The data is roughly a straight line through the origin.

```
map<double, double> my_data;
my_data[-1.1] = -1.2;
my_data[-2.3] = -2.1;
my_data[-2.9] = -3.3;
my_data[-4.1] = -4.3;
my_data[-5.0] = -5.3;
my_data[-6.1] = -5.9;
my_data[0.] = 0.;
my_data[1.1] = 1.2;
my_data[2.3] = 2.1;
my_data[2.9] = 3.3;
my_data[4.1] = 4.3;
my_data[5.0] = 5.3;
my_data[6.1] = 5.9;
```

First construct, size and draw a simple plot ready to add some sample lines.

```
svg_2d_plot my_plot;
my_plot.size(400, 400);
my_plot.plot(my_data, "my_data").fill_color(red);
```

First draw a line using SVG coordinates (rather than the Cartesian coordinates used for user's data - see below). Note that for SVG coordinates, Y increases **down** the page, so Y = 0 is the top and Y = 300 is the bottom. Default black is provided for color.

```
my_plot.draw_line(10, 390, 390, 10);
```

This line almost reaches the corners of the SVG image, but for plotting XY graphs, you probably don't want SVG coordinates, but want to use Cartesian coordinates for user's data. Now draw a blue line using the Cartesian coordinates for user's data, from the bottom left through the origin of the plot to the top right of the plot.

```
my_plot.draw_plot_line(-10, -10, +10, +10, blue);
```

If you have calculated a confidence interval, you might want to add curved line(s) showing it (still using the Cartesian coordinates). For example, you can draw a curve (quadratic) through two X, Y pairs using a Bezier curve with the middle point as control point.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
my_plot.draw_plot_curve(-6, -8, 0, +1, +8, +6, red);
```

Finally write the SVG image file.

> **Note**
>
> At present, notes and lines must come after all plot commands to be put in the correct place.

```
my_plot.write("./demo_2d_lines");

//show_2d_plot_settings(my_plot);
```

[demo_2d_lines_2]

The resulting plot is

See demo_2d_lines.cpp for full source code and sample output.

Histograms of 2D data

See demo_2d_histogram.cpp for full source code and sample output.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

The resulting histograms are

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Tutorial: Boxplot

Simple Example

```
#include <boost/svg_plot/svg_2d_plot.hpp>
using namespace boost::svg;
#include <vector>
using std::vector;

// Functions to simulate distributions.
double f(double x)
{
    return 50 / x;
}

double g(double x)
{
    return 40 + 25 * sin(x * 50);
}

int main()
{
    std::vector<double> data1, data2;

    // Fill our vectors with values from the functions f(x) and g(x)
    for(double i = .1; i < 10; i+=.1)
    {
        data1.push_back(f(i));
        data2.push_back(g(i));
    }

    // Initialize a new box plot.
    svg_boxplot my_plot;

    // Color information.
    my_plot.background_border_color(black);

    // String information.
    my_plot.title("Boxplots of Common Functions")
   .x_label("Functions")
   .y_label("Population Size");

    // Axis information.
    my_plot.y_range(0, 100)
           .y_minor_tick_length(20)
     .y_major_interval(20);

    // Write data.
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
    my_plot.plot(data1, "[50 / x]");
    my_plot.plot(data2, "[40 + 25 * sin(x * 50)]");

    my_plot.write("boxplot_simple.svg");
    return 0;
}
```

Image boxplot_simple.svg

Basic Example Breakdown

Let's examine what this does.

Initializing a new boxplot

This initializes a new boxplot. This also sets the many default values. 4 svg_boxplot my_plot;

Setting a color

This sets the border color of the entire image to black (not just the plot area).

```
my_plot.background_border_color(black);
```

Setting strings in the plot

The following code sets the title that appears at the top of the graph, the label that appears at the bottom to name the X-axis, and the label that appears to the left to name the Y-axis.

```
my_plot.title("Boxplots of Common Functions")
      .x_label("Functions")
      .y_label("Population Size");
```

Setting axis information

Axis information for the X-axis is very limited, as the coordinate system for the X-axis is discrete, based only on the number of box-and-whisker plots that appear in the same boxplot.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Axis information for the Y-axis, however, is customizable much in the same way that that axis information is customizable for the 1 and 2 dimensional graphs. // Axis information. my_plot.y_range(0, 100) .y_minor_tick_length(20) .y_major_interval(20);

Writing to a file

This finally writes our plot to the file "boxplot_simple.svg".

```
my_plot.write("boxplot_simple.svg");
```

Definitions of the Quartiles

This example demonstrates the dramatic effect of the choice of definition of the quartiles.

"Some Implementations of the Boxplot" Michael Frigge, David C. Hoaglin and Boris Iglewicz The American Statistician, Vol. 43, No. 1 (Feb., 1989), pp. 50-54 discusses the design of the boxplot.

However the plot of their example data shown below shows the considerable variation in the appearance of the same data, using different definitions of quartiles used in various popular statistics packages.

One obvious conclusion is that you should not expect boxplots to look the same when using more than one program.

Boost.Plot provides 5 popular definitions for the quartiles. This should allow the user to produce plots that look similar to boxplots from most statistics plotting program. To confuse matter further, most have their own default definition **and** options to chose other definitions: these options are shown below as type, method, PCTLDEF.

The interquartile range is calculated using the 1st & 3rd sample quartiles, but there are various ways to calculate those quartiles, summarised in Rob J. Hyndman and Yanan Fan, 1996, "Sample Quantiles in Statistical Packages", The American Statistician 50(4):361-365, (1996).

The interquartile range, often called IQR is quartile 3 (p = 3/4) - quartile 1 (1/4). The median is the 2nd quartile (p = 2/4 = 1/2).

Five of Hyndman and Fan's sample quantile definitions have a particularly simple common form selected according to which definition of m is chosen in function quantiles. This is implemented in function quantiles by parameter `HF_definition`:

```
double quantile(vector<double>& data, double p, int HF_definition = 8);
```

The default definition is that recommended by Hyndman and Fan, or users can select which definition is used for all boxplots, or individual data series as shown in the example below.

```
my_boxplot.quartile_definition(5); // All plots

my_boxplot.plot.quartile_definition(7); // Just this data series plot.
```

Hyndman and Fan definitions 4 to 8 are used by the following packages:

- #4 SAS (PCTLDEF=1), R (type=4), Maple (method=3)
- #5 R (type=5), Maple (method=4), Wolfram Mathematica quartiles.
- #6 Minitab, SPSS, BMDP, JMP, SAS (PCTLDEF=4), R(type=6), Maple (method=5).
- #7 Excel, S-Plus, R (type=7[default]), Maxima, Maple (method=6).
- #8 H&F 8: R (type=8), Maple (method=7[default]).

Some observations on the various options are:

- #4 Often a moderate interquartile range.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

- #5 Symmetric linear interpolation: a common choice when the data represent a sample from a continuous distribution and you want an unbiased estimate of the quartiles of that distribution.
- #6 This "half" sample excludes the sample median (k observations) for odd n (=2*k+1). This will tend to be a better estimate for the population quartiles, but will tend to give quartile estimates that are a bit too far from the center of the whole sample (too wide an interquartile range).
- #7 Smallest interquartile range, so flags most outliers. For a continuous distribution, this will tend to give too narrow an interquartile range, since there will tend to be a small fraction of the population beyond the extreme sample observations. In particular, for odd n (=2*k+1), Excel calculates the 1st (3rd) quartile as the median of the lower (upper) "half" of the sample including the sample median (k+1 observations).
- #8 recommended by H&F because it is approximately median-unbaised estimate regardless of distribution and thus suitable for continuous and discrete distributions. which gives quartiles between those reported by Minitab and Excel. This approach is approximately median unbiased for continuous distributions. Slightly higher interquartile range than definition 7.

The 'fences' beyond which points are regarded as outliers, or extreme outliers, are a multiplying factor, usually called k, and usually 1.5 * interquartile range, and 3 * interquartile range as recommended by Hoaglin et al.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
#include <vector>
using std::vector;
#include <cmath>
using ::sin;
//#include <boost/assert.hpp> // for BOOST_ASSERT
#include <boost/svg_plot/svg_boxplot.hpp>

#include <boost/svg_plot/quantile.hpp>
using boost::svg::quantile;

// double boost::svg::quantile(vector<double>& data, double p, int HF_definition);
// Estimate pth quantile of data using one of 5 definitions.
// Default HF_definition is the recommendation of Hyndman and Fan, definition #8.

#include <boost/array.hpp>
  using boost::array;

#include <iostream>
using std::cout;
using std::endl;
```

```
]
  // 11 values from Hoaglin et al page 50.
  const boost::ar↵
ray<double, 11> Hoaglin_data = {53., 56., 75., 81., 82., 85., 87., 89., 95., 99., 100.};
  //                                                        q1           median           q3

  vector<double> Hoaglin(Hoaglin_data.begin(), Hoaglin_data.end());
  for (int def = 4; def <= 8; def++)
  { // All the F&Y definitions of quartiles.
    double q1 = quantile(Hoaglin, 0.25, def); // 75
    double q2 = quantile(Hoaglin, 0.5, def); // 85
    double q3 = quantile(Hoaglin, 0.75, def); // 95
    cout << "Hoaglin definition #" << def << ", q1 " << q1
      << ", q2 " << q2 << ", q3 " << q3 << ", IQR " << q3 - q1 << endl;
  } // for

  // Same data copied for different data series.
  vector<double> Hoaglin4(Hoaglin_data.begin(), Hoaglin_data.end());
  vector<double> Hoaglin5(Hoaglin_data.begin(), Hoaglin_data.end());
  vector<double> Hoaglin6(Hoaglin_data.begin(), Hoaglin_data.end());
  vector<double> Hoaglin7(Hoaglin_data.begin(), Hoaglin_data.end());
  vector<double> Hoaglin8(Hoaglin_data.begin(), Hoaglin_data.end());

  svg_boxplot H_boxplot;
```

Show the quartile definition default.

```
cout << "Default boxplot.quartile_definition() = " << H_boxplot.quartile_definition() << endl; // 8
```

Add title, labels, range etc to the whole boxplot:

```
H_boxplot  // Title and axes labels.
  .title("Hoaglin Example Data")
  .x_label("Boxplot")
  .y_label("Value")
  .y_range(45, 115)  // Y-Axis range.
  .y_minor_tick_length(2)
  .y_major_interval(10);
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Add a few setting to the plot including setting quartile definition (though is actually same as the default 8), and show that the value is stored.

```
svg_boxplot& b = H_boxplot.median_values_on(true)
.outlier_values_on(true)
.extreme_outlier_values_on(true)
.quartile_definition(8)
;
```

Show the quartile definition just assigned:

```
cout << "boxplot.quartile_definition() = " << b.quartile_definition() << endl; // 8
```

Add a data series container, and labels, to the plot using the whole boxplot quartile definition set.

```
H_boxplot.plot(Hoaglin_data, "default_8");
```

Add another data series container, and the labels, to the plot, and select a **different** quartile definition.

```
svg_boxplot_series& d4 =
H_boxplot.plot(Hoaglin4, "def #4")
.whisker_length(4.)
.quartile_definition(4);
```

Show the quartile definition just assigned to the this data series.

```
cout << "boxplot_series.quartile_definition() = " << d4.quartile_definition() << endl; // 4
```

Add yet more data series container, and the labels, to the plot, and select a **different** quartile definition for each.

```
H_boxplot.plot(Hoaglin5, "def #5")
.whisker_length(5.)
.quartile_definition(5);

H_boxplot.plot(Hoaglin6, "def #6")
.whisker_length(6.)
.quartile_definition(6);

H_boxplot.plot(Hoaglin6, "def #7")
.whisker_length(7.)
.quartile_definition(7);

H_boxplot.plot(Hoaglin6, "def #8")
.whisker_length(8.)
.quartile_definition(8);
```

Write the entire SVG plot to a file.

```
H_boxplot.write("demo_Hoaglin.svg");
```

You can view the plot at demo_Hoaglin.svg.

Typical output is:

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
Autorun "j:\Cpp\SVG\debug\demo_Hoaglin.exe"
Hoaglin definition #4, q1 70.25, q2 83.5, q3 90.5, IQR 20.25
Hoaglin definition #5, q1 76.5, q2 85, q3 93.5, IQR 17
Hoaglin definition #6, q1 75, q2 85, q3 95, IQR 20
Hoaglin definition #7, q1 78, q2 85, q3 92, IQR 14
Hoaglin definition #8, q1 76, q2 85, q3 94, IQR 18
Default boxplot.quartile_definition() = 8
boxplot.quartile_definition() = 8
boxplot_series.quartile_definition() = 4
```

See demo_Hoaglin.cpp for full source code and sample output.

SVG tutorial

The SVG interface is only documented in the reference section.

Most users will want to use the plot interfaces.

A few very rudimentary examples of use (mainly historically used to testbed various features used in the plot interfaces) can be seen at

../../example/demo_svg.cpp

Showing data values at Numerical Limits

All limits that are dealt with are double precision floating-point limits. Data values that are 'at limits' are detected when data is added to the plot by the call to the `plot()` method.

Currently, the line interpolation algorithms do not take limits into account, so behavior may be incorrect for such plots as `1 / x`.

Data at the limits are drawn, by default, as a cone with stroke_color is `lightgray` and fill_color is `whitesmoke`.

This is customizable using, for example:

```
my_plot.limit_color(red).limit_fill_color(green);
```

NaN

Any double precision floating-point numbers that return a nonzero value for the function `_isnan(double)` is considered to be a NaN value. When plotted, the number will appear in the user-defined coordinates as 0.

Infinity

Any double precision floating point number that is equal to either of the following is considered to be Infinity:

```
std::numeric_limits<double>::max()
std::numeric_limits<double>::infinity()
```

When plotted, these values will appear at the **top** of your plot window. If the plot window is not turned on, these points will appear at the top of the image.

Negative Infinity

Any double precision floating point number that is equal to either of the following is considered to be Negative Infinity:

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
std::numeric_limits<double>::min()
-std::numeric_limits<double>::infinity()
std::numeric_limits<double>::denorm_min()
```

When plotted, these values will appear at the **bottom** of your plot window. If the plot window is not turned on, these points will appear at the bottom of the image.

Here is an example of showing values at numeric limits:

See also examples of limits in 1-D and 2-D plots. ../../example/demo_1d_limits.cpp ../../example/demo_2d_limits.cpp

Reference

Header <boost/svg_plot/detail/auto_axes.hpp>

Scalable Vector Graphic (SVG) autoscaling of axes.

Inspect container or data values to find minimum and maximum, avoiding values that are NaN and/or 'at limit'. scale axis using max and min values (calculated or user provided) optionally to include the orgin, and to set the ticks. Provide fine control over any overlap at the edges of the axes to avoid a tiny amount over the limit resulting in an ugly extra major tick. Also allow optional forcing of the ticks to be multiples of 1, 2, 5, 10.

7 Feb 09

Paul A. Bristow

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
namespace boost {
  namespace math {
    template<class FPT > bool isfinite(FPT);
  }
  namespace svg {
    template<typename iter > int mnmx(iter, iter, double *, double *);
    template<class T > std::pair< double, double > range(const T &);
    template<class T > std::pair< double, double > range_all(const T &);
    double rounddown10(double);
    double rounddown2(double);

    // Round value down to nearest multiple of 5.
    double rounddown5(double value);
    double roundup10(double);
    double roundup2(double);
    double roundup5(double);
    template<class T >
      void scale_axis(const T &, double *, double *, double *, int *,
                      double *, double *, double *, int *, bool = true,
                      double = 3., bool = false, double = 0., int = 6,
                      int = 0, bool = false, double = 0., int = 6, int = 0);
    template<class T >
      void scale_axis(const T &, double *, double *, double *, int *, bool,
                      double = 3., bool = false, double = 0., int = 6,
                      int = 0);
    template<typename iter >
      void scale_axis(iter, iter, double *, double *, double *, int *, bool,
                      double, bool = false, double = 0., int = 6, int = 0);

    // scale axis function to define axis marker ticks based on min & max parameters values.
    void scale_axis(double min_value, double max_value,
                    double * axis_min_value, double * axis_max_value,
                    double * axis_tick_increment, int * auto_ticks,
                    bool check_limits, double autoscale_plusminus,
                    bool origin, double tight, int min_ticks, int steps);

    // Scale axis and update min and max axis values, and tick increment and number of ticks.
    void scale_axis(double min_value, double max_value,
                    double * axis_min_value, double * axis_max_value,
                    double * axis_tick_increment, int * auto_ticks,
                    bool origin, double tight, int min_ticks, int steps);
    template<typename iter > size_t show(iter, iter);
    template<typename T > size_t show(const T &);
    template<typename T > size_t show_all(const T &);
  }
}
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Function template isfinite

boost::math::isfinite — If a floating-point value is finite, return true.

Synopsis

```
// In header: <boost/svg_plot/detail/auto_axes.hpp>


template<class FPT > bool isfinite(FPT t);
```

Description

Parameters: `t` floating-point value.
Returns: true if is finite, false if + or - infinite or any NaN.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Function template mnmx

boost::svg::mnmx — Round up and down to 2, 4, 6, 8, 10, or 5, 10 or 2, 5, 10 systems.

Synopsis

```
// In header: <boost/svg_plot/detail/auto_axes.hpp>


template<typename iter >
  int mnmx(iter begin, iter end, double * min, double * max);
```

Description

Inspect values to find min and max.

Inspect all values between begin and (before) end to work out and update min and max. Similar to boost::minmax_element, but ignoring at 'limit': non-finite, +-infinity, max & min, & NaN). If can't find a max and a min, then throw a runtime_error exception.

Returns: number of normal values (not NaN nor infinite).

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Function template range

boost::svg::range — Container Data series.

Synopsis

```
// In header: <boost/svg_plot/detail/auto_axes.hpp>


template<class T > std::pair< double, double > range(const T & container);
```

Description

Returns: minimum and maximum of an STL container.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Function template range_all

boost::svg::range_all

Synopsis

```
// In header: <boost/svg_plot/detail/auto_axes.hpp>


template<class T > std::pair< double, double > range_all(const T & containers);
```

Description

Returns: minimum and maximum of a container containing STL containers.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Function rounddown10

boost::svg::rounddown10 — Round value down to nearest multiple of 10.

Synopsis

```
// In header: <boost/svg_plot/detail/auto_axes.hpp>


double rounddown10(double value);
```

Description

Decimal scaling steps, so value is 0.1, 0.2, 0.5, 1., 2., 5. or 10., 10., 20., 100. ...

Decimal scaling steps, so value is 0.1, 0.2, 0.5, 1., 2., 5. or 10., 10., 20., 100. ...

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Function rounddown2

boost::svg::rounddown2 — Round value down to nearest multiple of 2.

Synopsis

```
// In header: <boost/svg_plot/detail/auto_axes.hpp>


double rounddown2(double value);
```

Description

Binary scaling steps, so value is 0.1, 0.2, 0.4, 0.6, 0.8, 0., 2., 4., 6., 8., 10., 20., 40., 60, 80, 100. ...

Binary scaling steps, so value is 0.1, 0.2, 0.4, 0.6, 0.8, 0., 2., 4., 6., 8., 10., 20., 40., 60, 80, 100. ...

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Function roundup10

boost::svg::roundup10 — Round value up to nearest multiple of 10.

Synopsis

```
// In header: <boost/svg_plot/detail/auto_axes.hpp>


double roundup10(double value);
```

Description

Decimal scaling steps, so value is 0.1, 0.2, 0.5, 1., 2., 5. or 10., 20., 50., 100. ...

Decimal scaling steps, so value is 0.1, 0.2, 0.5, 1., 2., 5. or 10., 20., 50., 100. ...

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Function roundup2

boost::svg::roundup2 — Round value up to nearest multiple of 2.

Synopsis

```
// In header: <boost/svg_plot/detail/auto_axes.hpp>


double roundup2(double value);
```

Description

Binary scaling steps, so return 0.1, 0.2, 0.4, 0.6, 0.8, 1.0, 2, 4, 6, 8, 10, 20, 40 60, 80, 100...

Binary scaling steps, so return 0.1, 0.2, 0.4, 0.6, 0.8, 1.0, 2, 4, 6, 8, 10, 20, 40 60, 80, 100...

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Function roundup5

boost::svg::roundup5 — Round value up to nearest multiple of 5.

Synopsis

```
// In header: <boost/svg_plot/detail/auto_axes.hpp>


double roundup5(double value);
```

Description

Binary scaling, so return 0.1, 0.5, 1, 5, 10, 50, 100 ....

Binary scaling, so return 0.1, 0.5, 1, 5, 10, 50, 100 ....

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Function template scale_axis

boost::svg::scale_axis — Scale X and Y axis using T a 2D STL container: array of pairs, vector of pairs, ...

Synopsis

```
// In header: <boost/svg_plot/detail/auto_axes.hpp>


template<class T >
  void scale_axis(const T & container, double * x_axis_min_value,
                  double * x_axis_max_value, double * x_axis_tick_increment,
                  int * x_auto_ticks, double * y_axis_min_value,
                  double * y_axis_max_value, double * y_axis_tick_increment,
                  int * y_auto_ticks, bool check_limits = true,
                  double autoscale_plusminus = 3., bool x_origin = false,
                  double x_tight = 0., int x_min_ticks = 6, int x_steps = 0,
                  bool y_origin = false, double y_tight = 0.,
                  int y_min_ticks = 6, int y_steps = 0);
```

Description

<

Scale X and Y axis using T a 2D STL container: array, vector, list, map ...

Parameters: `container` Data series to plot - entire 2D container.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Function template scale_axis

boost::svg::scale_axis — Scale axis using all the 1D data in an STL container.

Synopsis

```
// In header: <boost/svg_plot/detail/auto_axes.hpp>


template<class T >
  void scale_axis(const T & container, double * axis_min_value,
                  double * axis_max_value, double * axis_tick_increment,
                  int * auto_ticks, bool check_limits,
                  double autoscale_plusminus = 3., bool origin = false,
                  double tight = 0., int min_ticks = 6, int steps = 0);
```

Description

scale axis using an Entire Container Data series, usually to plot. (not necessarily ordered, so will find min and max).

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Function template scale_axis

boost::svg::scale_axis — Scale axis using an iterator into an STL container.

Synopsis

```
// In header: <boost/svg_plot/detail/auto_axes.hpp>


template<typename iter >
  void scale_axis(iter begin, iter end, double * axis_min_value,
                  double * axis_max_value, double * axis_tick_increment,
                  int * auto_ticks, bool check_limits,
                  double autoscale_plusminus, bool origin = false,
                  double tight = 0., int min_ticks = 6, int steps = 0);
```

Description

Scale axis from data series (usually to plot), perhaps only part of container.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Function template show

boost::svg::show

Synopsis

```
// In header: <boost/svg_plot/detail/auto_axes.hpp>


template<typename iter > size_t show(iter begin, iter end);
```

Description

Utility function to display STL containers.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Function template show

boost::svg::show

Synopsis

```
// In header: <boost/svg_plot/detail/auto_axes.hpp>


template<typename T > size_t show(const T & container);
```

Description

Utility functios to display STL containers.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Function template show_all

boost::svg::show_all

Synopsis

```
// In header: <boost/svg_plot/detail/auto_axes.hpp>


template<typename T > size_t show_all(const T & containers);
```

Description

Show all the containers values.

Header <boost/svg_plot/detail/axis_plot_frame.hpp>

SVG Plot functions common to 1D, 2D and Boxplots.

Functions are derived from base class axis_plot_frame.

Mar 2009

Jacob Voytko and Paul A. Bristow

```
namespace boost {
  namespace svg {

    // Placing of legend box, if requested by legend_on == true.
    enum legend_places { nowhere =  0, inside =  -1, outside_left =  1,
                         outside_right =  +2, outside_top =  +3,
                         outside_bottom =  +4, somewhere =  +5 };

    // If and how the X axes intersects Y axis.
    enum x_axis_intersect { bottom =  -1, x_intersects_y =  0, top =  +1 };

    // If and how the Y axes intersects X axis.
    enum y_axis_intersect { left =  -1, y_intersects_x =  0, right =  +1 };

    static const double reducer;    // To make uncertainty and degrees of freedom estimates a ↵
bit smaller font to help distinguish from value.
    static const double sin45;    // Use to calculate 'length' if axis value labels are sloping.
  }
}
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Global reducer

boost::svg::reducer — To make uncertainty and degrees of freedom estimates a bit smaller font to help distinguish from value.

Synopsis

```
// In header: <boost/svg_plot/detail/axis_plot_frame.hpp>

static const double reducer;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Global sin45

boost::svg::sin45 — Use to calculate 'length' if axis value labels are sloping.

Synopsis

```
// In header: <boost/svg_plot/detail/axis_plot_frame.hpp>

static const double sin45;
```

Header <boost/svg_plot/detail/FP_compare.hpp>

```
template<typename FPT  = double> class close_to;
template<typename FPT  = double> class smallest;

enum floating_point_comparison_type { FPC_STRONG, FPC_WEAK };

typedef close_to< double > neareq;
typedef smallest< double > tiny;
template<class T > T epsilon(T);
template<typename FPT > FPT fpt_abs(FPT arg);
template<class T > T max_value(T);
template<class T > T min_value(T);
template<typename FPT > FPT safe_fpt_division(FPT f1, FPT f2);
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Class template close_to

close_to

Synopsis

```
// In header: <boost/svg_plot/detail/FP_compare.hpp>

template<typename FPT  = double>
class close_to {
public:
  // construct/copy/destruct
  close_to();
  template<typename FPT >
    close_to(FPT, floating_point_comparison_type = FPC_STRONG);

  // public member functions
  bool operator()(FPT, FPT) const;
  FPT size() ;
  floating_point_comparison_type strength() ;
  FPT fraction_tolerance_;
  floating_point_comparison_type strong_or_weak_;
};
```

Description

`close_to` public construct/copy/destruct

1.
```
close_to();
```

2.
```
template<typename FPT >
  close_to(FPT tolerance,
           floating_point_comparison_type fpc_type = FPC_STRONG);
```

`close_to` public member functions

1.
```
bool operator()(FPT left, FPT right) const;
```

2.
```
FPT size() ;
```

3.
```
floating_point_comparison_type strength() ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Class template smallest

smallest

Synopsis

```
// In header: <boost/svg_plot/detail/FP_compare.hpp>

template<typename FPT  = double>
class smallest {
public:
  // construct/copy/destruct
  smallest();
  template<typename FPT > smallest(FPT);

  // public member functions
  template<typename FPT > bool operator()(FPT) ;
  template<typename FPT > bool operator()(FPT, FPT) ;
  FPT size() ;
  FPT smallest_;
};
```

Description

`smallest` public construct/copy/destruct

1.
```
smallest();
```

2.
```
template<typename FPT > smallest(FPT s);
```

`smallest` public member functions

1.
```
template<typename FPT > bool operator()(FPT fp_value) ;
```

2.
```
template<typename FPT > bool operator()(FPT fp_value, FPT s) ;
```

3.
```
FPT size() ;
```

Header <boost/svg_plot/detail/functors.hpp>

Functors to convert data to doubles.

SVG plot assumes all data are convertible to double or uncertain value type unc before being plotted. The functors are used to convert both 1D and 2D (pairs of data values) to be converted. Note that uncertain value class unc only holds double precision and long double data will therefore lose information. This seems reasonable design decision as any real data to be plotted is unlikely to have more than double precision (about 16 decimal digits).

Jacob Voytko and Paul A. Bristow

Mar 2009

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Header <boost/svg_plot/detail/numeric_limits_handling.hpp>

Functions to check if data values are NaN or infinity or denormalised.

Since only double is used, template versions are not needed, and TR1 should provide max, min, denorm_min, infinity and isnan, but older compilers and libraries may not provide all these.

Jacob Voytko and Paul A. Bristow

Header <boost/svg_plot/detail/pair.hpp>

Provides a private implementation of operator<< for std::pair that outputs pairs with a comma separated format, for example: 1.2, 3.4.

Hidden in namespace detail to avoid clashes with other implementations of std::pair operator<<.

Mar 2009

Paul A. Bristow

Header <boost/svg_plot/detail/svg_boxplot_detail.hpp>

Boost.Plot SVG Box plot Implemention details.

See svg_boxplot.hpp for user functions. See also svg_style_detail.hpp for enum plot_doc_structure. Caution: these two enum and ids must match because the enum value is used to index the array of id strings. void set_ids() copies all strings to matching image.get_g_element(i).id() So add any new id items to both!

Jacob Voytko and Paul A. Bristow

Mar 2009

```
namespace boost {
  namespace svg {
    namespace boxplot {

     // groups that form the boxplot svg document structure. Order controls the painting order, ↵
later ones overwriting earlier layers.
      enum boxplot_doc_structure { PLOT_BACKGROUND =  0,
                                   PLOT_WINDOW_BACKGROUND, X_AXIS, Y_AXIS,
                                   X_TICKS, Y_MAJOR_TICKS, Y_MINOR_TICKS,
                                   Y_MAJOR_GRID, Y_MINOR_GRID, VALUE_LABELS,
                                   Y_LABEL, X_LABEL, BOX_AXIS, BOX, MEDIAN,
                                   WHISKER, MILD_OUTLIERS, EXTREME_OUTLIERS,
                                   DATA_VALUE_LABELS, PLOT_TITLE, PLOT_NOTES,
                                   BOXPLOT_DOC_CHILDREN };

      std::string document_ids_;      // String descriptors used in SVG XML (matching enum box↵
plot_doc_structure).
    }
  }
}
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Global document_ids_

boost::svg::boxplot::document_ids_ — String descriptors used in SVG XML (matching enum boxplot_doc_structure).

Synopsis

```
// In header: <boost/svg_plot/detail/svg_boxplot_detail.hpp>

std::string document_ids_;
```

Header <boost/svg_plot/detail/svg_style_detail.hpp>

Plot document structure whose order controls the painting order, later layers overwriting earlier layers.

Jul 2009

Jacob Voytko and Paul A. Bristow

Header <boost/svg_plot/detail/svg_tag.hpp>

Boost.Plot SVG plot Implemention details.

See svg.hpp etc for user functions. svg_tag.hpp defines all classes that can occur in the SVG parse tree.

Jacob Voytko and Paul A. Bristow

Mar 2009

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
namespace boost {
  namespace svg {
    struct a_path;
    struct c_path;

    class circle_element;
    class clip_path_element;
    class ellipse_element;
    class g_element;

    struct h_path;
    struct l_path;

    class line_element;

    struct m_path;

    class path_element;

    struct path_point;
    struct poly_path_point;
    struct polygon_element;

    class polyline_element;

    struct q_path;

    class qurve_element;
    class rect_element;

    struct s_path;

    class svg_element;

    struct t_path;

    class text_element;
    class text_element_text;
    class text_parent;
    class tspan_element;

    struct v_path;
    struct z_path;

    // Represents a single block of text, with font & alignment.
    enum align_style { left_align, right_align, center_align };
    bool operator!=(const rect_element &, const rect_element &);
    std::ostream & operator<<(std::ostream &, polyline_element &);
    std::ostream & operator<<(std::ostream &, polygon_element &);
    std::ostream & operator<<(std::ostream &, const poly_path_point &);
    std::ostream & operator<<(std::ostream &, text_element &);
    std::ostream & operator<<(std::ostream &, const rect_element &);
    bool operator==(const rect_element &, const rect_element &);
  }
}
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Struct a_path

boost::svg::a_path — Draws a elliptical arc from the current point to (x,y), using two radii, axis rotation, and control two flags.

Synopsis

```
// In header: <boost/svg_plot/detail/svg_tag.hpp>


struct a_path : public boost::svg::path_point {
  // construct/copy/destruct
  a_path(double, double, double, double, double, bool = false, bool = false,
         bool = false);

  // public member functions
  void write(std::ostream &) ;
  bool large_arc;  // true if arc >= 180 degrees wanted.
  bool relative;  // If true relative else absolute.
  double rx;  // X radius.
  double ry;  // Y radius.
  bool sweep;  // true if to draw in positive-angle direction
  double x;  // X End of arc from current point.
  double x_axis_rotation;  // Any rotation of the X axis.
  double y;  // Y End of arc from current point.
};
```

Description

See 8.3.8 The elliptical arc curve commands.! Useful for pie charts, etc.

`a_path` public construct/copy/destruct

1.
```
a_path(double x, double y, double rx, double ry, double x_axis_rotation,
       bool large_arc = false, bool sweep = false, bool relative = false);
```

   Construct elliptic arc path.

`a_path` public member functions

1.
```
void write(std::ostream & o_str) ;
```

   Write elliptical arc path XML to ostream.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Struct c_path

boost::svg::c_path — Draws a cubic Bezier curve from the current point to (x, y) using (x1, y1).

Synopsis

```
// In header: <boost/svg_plot/detail/svg_tag.hpp>


struct c_path : public boost::svg::path_point {
  // construct/copy/destruct
  c_path(double, double, double, double, double, double, bool = false);

  // public member functions
  void write(std::ostream &) ;
  bool relative;  // If true relative else absolute.
  double x;  // Current (start point).
  double x1;  // Middle of curve.
  double x2;  // End point.
  double y;  // Current (start point).
  double y1;  // Middle of curve.
  double y2;  // End point.
};
```

Description

8.3.5 The curve commands: C, Q & A.

`c_path` public construct/copy/destruct

1.
```
c_path(double x1, double y1, double x2, double y2, double x, double y,
       bool relative = false);
```

`c_path` public member functions

1.
```
void write(std::ostream & o_str) ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Class circle_element

boost::svg::circle_element — Circle from center coordinate, and radius.

Synopsis

```
// In header: <boost/svg_plot/detail/svg_tag.hpp>


class circle_element : public boost::svg::svg_element {
public:
  // construct/copy/destruct
  circle_element(double, double, double, const svg_style &,
                 const std::string & = "", const std::string & = "",
                 const std::string & = "");
  circle_element(double, double, double = 5);

  // public member functions
  std::string class_id() ;
  void class_id(const std::string &) ;
  std::string clip_id() ;
  void clip_id(const std::string &) ;
  std::string id() ;
  void id(const std::string &) ;
  bool operator!=(const svg_element &) ;
  bool operator==(const svg_element &) ;
  const svg_style & style() const;
  svg_style & style() ;
  void write(std::ostream &) ;

  // protected member functions
  void write_attributes(std::ostream &) ;
  double radius;
  double x;
  double y;

  std::string class_name_;  // SVG id name, set & get by function id.
  std::string clip_name_;  // SVG class name, set & get by function class id.
  std::string id_name_;  // Colors fill, stroke, width, get by function style.
  svg_style style_info_;
};
```

Description

Represents a single circle. http://www.w3.org/TR/SVG/shapes.html#CircleElement

`circle_element` public construct/copy/destruct

1.
```
circle_element(double x, double y, double radius,
               const svg_style & style_info,
               const std::string & id_name = "",
               const std::string & class_name = "",
               const std::string & clip_name = "");
```

   Constructor defines all private data.

2.
```
circle_element(double x, double y, double radius = 5);
```

   Constructor defines all private data (default radius only).

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

`circle_element` public member functions

1. ```
   std::string class_id() ;
   ```

   Class id, non-unique identifier for an element.

2. ```
   void class_id(const std::string & class_id) ;
   ```

   Class class id, non-unique identifier for an element.

   http://www.w3.org/TR/2001/REC-SVG-20010904/styling.html#ClassAttribute 6.12 Attributes common to all elements: id and xml:base Example: class="info"

3. ```
   std::string clip_id() ;
   ```

   Returns: name of a clip path, for example: g_ptr.clip_id(plot_window_clip_);

4. ```
   void clip_id(const std::string & id) ;
   ```

   Set name of a clip path, for example: g_ptr.clip_id(plot_window_clip_);

5. ```
   std::string id() ;
   ```

   Returns: the unique name for an element, for example id() ="plotBackground".

6. ```
   void id(const std::string & id) ;
   ```

   Provide a unique name for an element.

   See http://www.w3.org/TR/SVG/struct.html#IDAttribute 5.10.1 Attributes common to all elements: id and xml:base The id and xml:base attributes are available on all SVG elements: Attribute definitions: id = "name" Standard XML attribute for assigning a unique name to an element. Refer to the "Extensible Markup Language (XML) 1.0" Recommendation [XML10]. xml:base = "<uri>" Specifies a base URI other than the base URI of the document or external entity. Refer to the "XML Base" specification [XML-BASE]. A group of elements, as well as individual objects, can be given a name using the id attribute. Named groups are needed for several purposes such as animation and re-usable objects. Example: id="plotBackground"

7. ```
   bool operator!=(const svg_element & lhs) ;
   ```

   Compare svg_elements for inequality, useful for Boost.Test.

8. ```
   bool operator==(const svg_element & lhs) ;
   ```

   Compare svg_elements, useful for Boost.Test.

9. ```
   const svg_style & style() const;
   ```

   Returns: reference to const svg_style to provide indirect access to colors & width via style().stroke_color(), fill_color(), width() (const version).

10. ```
    svg_style & style() ;
    ```

    Returns: reference to svg_style to provide indirect access to colors & width via style().stroke_color(), fill_color(), width()

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

11.
```
void write(std::ostream & rhs) ;
```

Output SVG XML Example: <circle cx="9.78571" cy="185" r="5"/>

`circle_element` protected member functions

1.
```
void write_attributes(std::ostream & s_out) ;
```

SVG clip path name, set & get by function clip_id.

Output group_element id and clip-path.

Classes inherited from svg_element add other references, 5.3.1, like color, fill, stroke, gradients... Example id: <g id="yMinorGrid"></g> Example class: <g class="grid_style"></g> Example URI: fill="url(#Gradient01) // local URL

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Class clip_path_element

boost::svg::clip_path_element — The clipping path restricts the region to which paint can be applied.

Synopsis

```
// In header: <boost/svg_plot/detail/svg_tag.hpp>


class clip_path_element : public boost::svg::svg_element {
public:
  // construct/copy/destruct
  clip_path_element(const std::string &, const rect_element &);

  // public member functions
  std::string class_id() ;
  void class_id(const std::string &) ;
  std::string clip_id() ;
  void clip_id(const std::string &) ;
  std::string id() ;
  void id(const std::string &) ;
  bool operator!=(const svg_element &) ;
  bool operator==(const svg_element &) ;
  const svg_style & style() const;
  svg_style & style() ;
  void write(std::ostream &) ;

  // protected member functions
  void write_attributes(std::ostream &) ;
  std::string element_id;
  rect_element rect;

  std::string class_name_;  // SVG id name, set & get by function id.
  std::string clip_name_;  // SVG class name, set & get by function class id.
  std::string id_name_;  // Colors fill, stroke, width, get by function style.
  svg_style style_info_;
};
```

Description

14.3 Clipping paths http://www.w3.org/TR/SVG/masking.html#ClipPathProperty.

`clip_path_element` public construct/copy/destruct

1. ```
   clip_path_element(const std::string & id, const rect_element & rect);
   ```

   Clipping rectangle.

`clip_path_element` public member functions

1. ```
   std::string class_id() ;
   ```

   Class id, non-unique identifier for an element.

2. ```
   void class_id(const std::string & class_id) ;
   ```

   Class class id, non-unique identifier for an element.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/
   ```

http://www.w3.org/TR/2001/REC-SVG-20010904/styling.html#ClassAttribute 6.12 Attributes common to all elements: id and xml:base Example: class="info"

3.
```
std::string clip_id() ;
```

Returns: name of a clip path, for example: g_ptr.clip_id(plot_window_clip_);

4.
```
void clip_id(const std::string & id) ;
```

Set name of a clip path, for example: g_ptr.clip_id(plot_window_clip_);

5.
```
std::string id() ;
```

Returns: the unique name for an element, for example id() ="plotBackground".

6.
```
void id(const std::string & id) ;
```

Provide a unique name for an element.

See http://www.w3.org/TR/SVG/struct.html#IDAttribute 5.10.1 Attributes common to all elements: id and xml:base The id and xml:base attributes are available on all SVG elements: Attribute definitions: id = "name" Standard XML attribute for assigning a unique name to an element. Refer to the "Extensible Markup Language (XML) 1.0" Recommendation [XML10]. xml:base = "<uri>" Specifies a base URI other than the base URI of the document or external entity. Refer to the "XML Base" specification [XML-BASE]. A group of elements, as well as individual objects, can be given a name using the id attribute. Named groups are needed for several purposes such as animation and re-usable objects. Example: id="plotBackground"

7.
```
bool operator!=(const svg_element & lhs) ;
```

Compare svg_elements for inequality, useful for Boost.Test.

8.
```
bool operator==(const svg_element & lhs) ;
```

Compare svg_elements, useful for Boost.Test.

9.
```
const svg_style & style() const;
```

Returns: reference to const svg_style to provide indirect access to colors & width via style().stroke_color(), fill_color(), width() (const version).

10.
```
svg_style & style() ;
```

Returns: reference to svg_style to provide indirect access to colors & width via style().stroke_color(), fill_color(), width()

11.
```
void write(std::ostream & os) ;
```

`clip_path_element` protected member functions

1.
```
void write_attributes(std::ostream & s_out) ;
```

SVG clip path name, set & get by function clip_id.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Output group_element id and clip-path.

Classes inherited from svg_element add other references, 5.3.1, like color, fill, stroke, gradients... Example id: <g id="yMinorGrid"></g> Example class: <g class="grid_style"></g> Example URI: fill="url(#Gradient01) // local URL

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Class ellipse_element

boost::svg::ellipse_element — Ellipse from center coordinate, and radius.

Synopsis

```
// In header: <boost/svg_plot/detail/svg_tag.hpp>


class ellipse_element : public boost::svg::svg_element {
public:
  // construct/copy/destruct
  ellipse_element(double, double, const svg_style &, const std::string & = "",
                  const std::string & = "", const std::string & = "");
  ellipse_element(double, double, double, double, const svg_style &,
                  const std::string & = "", const std::string & = "",
                  const std::string & = "");
  ellipse_element(double, double, double = 4, double = 8);

  // public member functions
  std::string class_id() ;
  void class_id(const std::string &) ;
  std::string clip_id() ;
  void clip_id(const std::string &) ;
  std::string id() ;
  void id(const std::string &) ;
  bool operator!=(const svg_element &) ;
  bool operator==(const svg_element &) ;
  const svg_style & style() const;
  svg_style & style() ;
  void write(std::ostream &) ;

  // protected member functions
  void write_attributes(std::ostream &) ;
  double cx_;  // coordinate x of center of ellipse, default 0.
  double cy_;  // coordinate y, default 0.
  double rotate_;
  double rx_;  // radius x.
  double ry_;  // radius y.

  std::string class_name_;  // SVG id name, set & get by function id.
  std::string clip_name_;  // SVG class name, set & get by function class id.
  std::string id_name_;  // Colors fill, stroke, width, get by function style.
  svg_style style_info_;
};
```

Description

Represents a single ellipse. http://www.w3.org/TR/SVG/shapes.html#EllipseElement 9.4 The 'ellipse' element. Default is 'horizontal' but can be rotated.

`ellipse_element` public construct/copy/destruct

1.
```
ellipse_element(double cx, double cy, const svg_style & style_info,
                const std::string & id_name = "",
                const std::string & class_name = "",
                const std::string & clip_name = "");
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

2.
```
ellipse_element(double cx, double cy, double rx, double ry,
                const svg_style & style_info,
                const std::string & id_name = "",
                const std::string & class_name = "",
                const std::string & clip_name = "");
```

< Constructor defines all private data.

3.
```
ellipse_element(double cx, double cy, double rx = 4, double ry = 8);
```

rotation in degrees from horizontal (default 0.).

< Constructor defines all private data (with default radii).

`ellipse_element` public member functions

1.
```
std::string class_id() ;
```

Class id, non-unique identifier for an element.

2.
```
void class_id(const std::string & class_id) ;
```

Class class id, non-unique identifier for an element.

http://www.w3.org/TR/2001/REC-SVG-20010904/styling.html#ClassAttribute 6.12 Attributes common to all elements: id and xml:base Example: class="info"

3.
```
std::string clip_id() ;
```

Returns: name of a clip path, for example: g_ptr.clip_id(plot_window_clip_);

4.
```
void clip_id(const std::string & id) ;
```

Set name of a clip path, for example: g_ptr.clip_id(plot_window_clip_);

5.
```
std::string id() ;
```

Returns: the unique name for an element, for example id() ="plotBackground".

6.
```
void id(const std::string & id) ;
```

Provide a unique name for an element.

See http://www.w3.org/TR/SVG/struct.html#IDAttribute 5.10.1 Attributes common to all elements: id and xml:base The id and xml:base attributes are available on all SVG elements: Attribute definitions: id = "name" Standard XML attribute for assigning a unique name to an element. Refer to the "Extensible Markup Language (XML) 1.0" Recommendation [XML10]. xml:base = "<uri>" Specifies a base URI other than the base URI of the document or external entity. Refer to the "XML Base" specification [XML-BASE]. A group of elements, as well as individual objects, can be given a name using the id attribute. Named groups are needed for several purposes such as animation and re-usable objects. Example: id="plotBackground"

7.
```
bool operator!=(const svg_element & lhs) ;
```

Compare svg_elements for inequality, useful for Boost.Test.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

8.
```
bool operator==(const svg_element & lhs) ;
```

Compare svg_elements, useful for Boost.Test.

9.
```
const svg_style & style() const;
```

Returns: reference to const svg_style to provide indirect access to colors & width via style().stroke_color(), fill_color(), width() (const version).

10.
```
svg_style & style() ;
```

Returns: reference to svg_style to provide indirect access to colors & width via style().stroke_color(), fill_color(), width()

11.
```
void write(std::ostream & os) ;
```

Output SVG XML for ellipse. Example: <ellipse rx="250" ry="100" fill="red" />

`ellipse_element` protected member functions

1.
```
void write_attributes(std::ostream & s_out) ;
```

SVG clip path name, set & get by function clip_id.

Output group_element id and clip-path.

Classes inherited from svg_element add other references, 5.3.1, like color, fill, stroke, gradients... Example id: <g id="yMinorGrid"></g> Example class: <g class="grid_style"></g> Example URI: fill="url(#Gradient01) // local URL

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Class g_element

boost::svg::g_element — g_element (group element) is the node element of our document tree. 'g' element is a container element for grouping together <g> ... </g>

Synopsis

```
// In header: <boost/svg_plot/detail/svg_tag.hpp>


class g_element : public boost::svg::svg_element {
public:
  // construct/copy/destruct
  g_element();

  // public member functions
  g_element & add_g_element() ;
  circle_element & circle(double, double, unsigned int = 5) ;
  std::string class_id() ;
  void class_id(const std::string &) ;
  void clear() ;
  std::string clip_id() ;
  void clip_id(const std::string &) ;
  ellipse_element & ellipse(double, double, double, double) ;
  g_element & g(int) ;
  polygon_element &
  hexagon(double, double, double, double, double, double, double, double,
          double, double, double, double, bool = true) ;
  std::string id() ;
  void id(const std::string &) ;
  line_element & line(double, double, double, double) ;
  bool operator!=(const svg_element &) ;
  bool operator==(const svg_element &) ;
  svg_element & operator[](unsigned int) ;
  path_element & path() ;
  polygon_element &
  pentagon(double, double, double, double, double, double, double, double,
           double, double, bool = true) ;
  polygon_element & polygon() ;
  polygon_element & polygon(std::vector< poly_path_point > &, bool = true) ;
  polygon_element & polygon(double, double, bool = true) ;
  polyline_element & polyline() ;
  polyline_element & polyline(double, double) ;
  polyline_element & polyline(std::vector< poly_path_point > &) ;
  void push_back(svg_element *) ;
  rect_element & rect(double, double, double, double) ;
  polygon_element &
  rhombus(double, double, double, double, double, double, double, double,
          bool = true) ;
  size_t size() ;
  const svg_style & style() const;
  svg_style & style() ;
  text_element &
  text(double = 0., double = 0., const std::string & = "",
       const text_style & = no_style, const align_style & = left_align,
       const rotate_style & = horizontal) ;
  polygon_element &
  triangle(double, double, double, double, double, double, bool = true) ;
  void write(std::ostream &) ;

  // protected member functions
  void write_attributes(std::ostream &) ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
  ptr_vector< svg_element > children;
  std::string clip_name;
  bool clip_on;

  std::string class_name_;  // SVG id name, set & get by function id.
  std::string clip_name_;  // SVG class name, set & get by function class id.
  std::string id_name_;  // Colors fill, stroke, width, get by function style.
  svg_style style_info_;
};
```

Description

g_element ('g' element is a container element for grouping together related graphics elements). See http://www.w3.org/TR/SVG/struct.html#NewDocument 5.2.1 Overview.

'g' element is a container element for grouping together <g /> </g> related graphics elements, for example, an image background rectangle with border and fill: <g id="background" fill="rgb(255,255,255)"><rect width="500" height="350"/></g> .

`g_element` public construct/copy/destruct

1.
```
g_element();
```

`g_element` public member functions

1.
```
g_element & add_g_element() ;
```

Add a new group element.
Returns: A reference to the new child node just created.

2.
```
circle_element & circle(double x, double y, unsigned int radius = 5) ;
```

Add a new circle element.
Returns: A reference to the new child node just created.

3.
```
std::string class_id() ;
```

Class id, non-unique identifier for an element.

4.
```
void class_id(const std::string & class_id) ;
```

Class class id, non-unique identifier for an element.

http://www.w3.org/TR/2001/REC-SVG-20010904/styling.html#ClassAttribute 6.12 Attributes common to all elements: id and xml:base Example: class="info"

5.
```
void clear() ;
```

Remove all the child nodes.

6.
```
std::string clip_id() ;
```

Returns: name of a clip path, for example: g_ptr.clip_id(plot_window_clip_);

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

7.
```
void clip_id(const std::string & id) ;
```

Set name of a clip path, for example: g_ptr.clip_id(plot_window_clip_);

8.
```
ellipse_element & ellipse(double rx, double ry, double cx, double cy) ;
```

Add a new ellipse element.
Returns: A reference to the new child node just created.

9.
```
g_element & g(int i) ;
```

i is index of children nodes.

10.
```
polygon_element &
hexagon(double x1, double y1, double x2, double y2, double x3, double y3,
        double x4, double y4, double x5, double y5, double x6, double y6,
        bool f = true) ;
```

Add a new hexagon element.
Returns: A reference to the new child node just created.

11.
```
std::string id() ;
```

Returns: the unique name for an element, for example id() ="plotBackground".

12.
```
void id(const std::string & id) ;
```

Provide a unique name for an element.

See http://www.w3.org/TR/SVG/struct.html#IDAttribute 5.10.1 Attributes common to all elements: id and xml:base The id and xml:base attributes are available on all SVG elements: Attribute definitions: id = "name" Standard XML attribute for assigning a unique name to an element. Refer to the "Extensible Markup Language (XML) 1.0" Recommendation [XML10]. xml:base = "<uri>" Specifies a base URI other than the base URI of the document or external entity. Refer to the "XML Base" specification [XML-BASE]. A group of elements, as well as individual objects, can be given a name using the id attribute. Named groups are needed for several purposes such as animation and re-usable objects. Example: id="plotBackground"

13.
```
line_element & line(double x1, double y1, double x2, double y2) ;
```

Add a new line element.
Returns: A reference to the new child node just created.

14.
```
bool operator!=(const svg_element & lhs) ;
```

Compare svg_elements for inequality, useful for Boost.Test.

15.
```
bool operator==(const svg_element & lhs) ;
```

Compare svg_elements, useful for Boost.Test.

16.
```
svg_element & operator[](unsigned int i) ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

17.
```
path_element & path() ;
```

Add a new path element.
Returns: A reference to the new path just created.

18.
```
polygon_element &
pentagon(double x1, double y1, double x2, double y2, double x3, double y3,
         double x4, double y4, double x5, double y5, bool f = true) ;
```

Add a new pentagon element.
Returns: A reference to the new child node just created.

19.
```
polygon_element & polygon() ;
```

Add a new polygon element.
Returns: A reference to the new polygon element just created.

20.
```
polygon_element & polygon(std::vector< poly_path_point > & v, bool f = true) ;
```

Add a new complete polygon element.
Returns: A reference to the new child node just created.// push_back a complete many-sided polygon to the document.

21.
```
polygon_element & polygon(double x, double y, bool f = true) ;
```

Add a new polygon element.
Returns: A reference to the new child node just created.

22.
```
polyline_element & polyline() ;
```

Add a new polyline element.
Returns: A reference to the new polyline element just created.

23.
```
polyline_element & polyline(double x, double y) ;
```

Add a new polyline element, but 1st point only, add others later with .P(x, y)...
Returns: A reference to the new child node just created.

24.
```
polyline_element & polyline(std::vector< poly_path_point > & v) ;
```

Add a new complete polyline.
Returns: A reference to the new child node just created.

25.
```
void push_back(svg_element * g) ;
```

Add a new child node g_element.

26.
```
rect_element & rect(double x1, double y1, double x2, double y2) ;
```

Add a new rect element.
Returns: A reference to the new child node just created.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

27.
```
polygon_element &
rhombus(double x1, double y1, double x2, double y2, double x3, double y3,
        double x4, double y4, bool f = true) ;
```

Add a new rhombus element.
Returns: A reference to the new child node just created.

28.
```
size_t size() ;
```

Returns: Number of child nodes.

29.
```
const svg_style & style() const;
```

Returns: reference to const svg_style to provide indirect access to colors & width via style().stroke_color(), fill_color(), width() (const version).

30.
```
svg_style & style() ;
```

Returns: reference to svg_style to provide indirect access to colors & width via style().stroke_color(), fill_color(), width()

31.
```
text_element &
text(double x = 0., double y = 0., const std::string & text = "",
     const text_style & style = no_style,
     const align_style & align = left_align,
     const rotate_style & rotate = horizontal) ;
```

Add a new text element.
Returns: A reference to the new child node just created.

32.
```
polygon_element &
triangle(double x1, double y1, double x2, double y2, double x3, double y3,
         bool f = true) ;
```

Add a new triangle element.
Returns: A reference to the new child node just created.

33.
```
void write(std::ostream & os) ;
```

Output all children of a group element. Example: <g fill="rgb(255,255,255)" id="background"><rect x="0" y="0" width="500" height="350"/></g>

Avoid useless output like: <g id="legendBackground"></g> TODO check this doesn't mean that useful style is lost?

`g_element` protected member functions

1.
```
void write_attributes(std::ostream & s_out) ;
```

SVG clip path name, set & get by function clip_id.

Output group_element id and clip-path.

Classes inherited from svg_element add other references, 5.3.1, like color, fill, stroke, gradients... Example id: <g id="yMinorGrid" ></g> Example class: <g class="grid_style"></g> Example URI: fill="url(#Gradient01) // local URL

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Struct h_path

boost::svg::h_path — Draws a horizontal line from the current point (cpx, cpy) to (x, cpy). which becomes the new current point. No y needed, start from current point y.

Synopsis

```
// In header: <boost/svg_plot/detail/svg_tag.hpp>


struct h_path : public boost::svg::path_point {
  // construct/copy/destruct
  h_path(double, bool = false);

  // public member functions
  void write(std::ostream &) ;
  bool relative;  // If true relative else absolute.
  double x;
};
```

Description

`h_path` public construct/copy/destruct

1.
```
h_path(double x, bool relative = false);
```

`h_path` public member functions

1.
```
void write(std::ostream & o_str) ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Struct l_path

boost::svg::l_path — Draw a line from the current point to the given (x,y) coordinate which becomes the new current point.

Synopsis

```
// In header: <boost/svg_plot/detail/svg_tag.hpp>


struct l_path : public boost::svg::path_point {
  // construct/copy/destruct
  l_path(double, double, bool = false);

  // public member functions
  void write(std::ostream &) ;
  bool relative;  // If true relative else absolute.
  double x;
  double y;
};
```

Description

`l_path` public construct/copy/destruct

1. ```
   l_path(double x, double y, bool relative = false);
   ```

`l_path` public member functions

1. ```
   void write(std::ostream & o_str) ;
   ```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Class line_element

boost::svg::line_element — Line from (x1, y1) to (x2, y2). /details Straight line from SVG location (x1, y1) to (x2, y2).

Synopsis

```
// In header: <boost/svg_plot/detail/svg_tag.hpp>


class line_element : public boost::svg::svg_element {
public:
  // construct/copy/destruct
  line_element(double, double, double, double, const svg_style &,
               const std::string & = "", const std::string & = "",
               const std::string & = "");
  line_element(double, double, double, double);

  // public member functions
  std::string class_id() ;
  void class_id(const std::string &) ;
  std::string clip_id() ;
  void clip_id(const std::string &) ;
  std::string id() ;
  void id(const std::string &) ;
  bool operator!=(const svg_element &) ;
  bool operator==(const svg_element &) ;
  const svg_style & style() const;
  svg_style & style() ;
  void write(std::ostream &) ;

  // protected member functions
  void write_attributes(std::ostream &) ;
  double x1_;  // Line from (x1_, x2_) to (y1_, y2_).
  double x2_;  // Line from (x1_, x2_) to (y1_, y2_).
  double y1_;  // Line from (x1_, x2_) to (y1_, y2_).
  double y2_;  // Line from (x1_, x2_) to (y1_, y2_).

  std::string class_name_;  // SVG id name, set & get by function id.
  std::string clip_name_;  // SVG class name, set & get by function class id.
  std::string id_name_;  // Colors fill, stroke, width, get by function style.
  svg_style style_info_;
};
```

Description

`line_element` public construct/copy/destruct

1.
```
line_element(double x1, double y1, double x2, double y2,
             const svg_style & style_info, const std::string & id_name = "",
             const std::string & class_name = "",
             const std::string & clip_name = "");
```

2.
```
line_element(double x1, double y1, double x2, double y2);
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

`line_element` public member functions

1. ```
   std::string class_id() ;
   ```

   Class id, non-unique identifier for an element.

2. ```
   void class_id(const std::string & class_id) ;
   ```

   Class class id, non-unique identifier for an element.

   http://www.w3.org/TR/2001/REC-SVG-20010904/styling.html#ClassAttribute 6.12 Attributes common to all elements: id and xml:base Example: class="info"

3. ```
   std::string clip_id() ;
   ```

   Returns: name of a clip path, for example: g_ptr.clip_id(plot_window_clip_);

4. ```
   void clip_id(const std::string & id) ;
   ```

   Set name of a clip path, for example: g_ptr.clip_id(plot_window_clip_);

5. ```
   std::string id() ;
   ```

   Returns: the unique name for an element, for example id() ="plotBackground".

6. ```
   void id(const std::string & id) ;
   ```

   Provide a unique name for an element.

   See http://www.w3.org/TR/SVG/struct.html#IDAttribute 5.10.1 Attributes common to all elements: id and xml:base The id and xml:base attributes are available on all SVG elements: Attribute definitions: id = "name" Standard XML attribute for assigning a unique name to an element. Refer to the "Extensible Markup Language (XML) 1.0" Recommendation [XML10]. xml:base = "<uri>" Specifies a base URI other than the base URI of the document or external entity. Refer to the "XML Base" specification [XML-BASE]. A group of elements, as well as individual objects, can be given a name using the id attribute. Named groups are needed for several purposes such as animation and re-usable objects. Example: id="plotBackground"

7. ```
   bool operator!=(const svg_element & lhs) ;
   ```

   Compare svg_elements for inequality, useful for Boost.Test.

8. ```
   bool operator==(const svg_element & lhs) ;
   ```

   Compare svg_elements, useful for Boost.Test.

9. ```
   const svg_style & style() const;
   ```

   Returns: reference to const svg_style to provide indirect access to colors & width via style().stroke_color(), fill_color(), width() (const version).

10. ```
    svg_style & style() ;
    ```

    Returns: reference to svg_style to provide indirect access to colors & width via style().stroke_color(), fill_color(), width()

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

11.
```
void write(std::ostream & rhs) ;
```

output line from (x1_, y1_) to (x2_, y2_) Write XML SVG command to draw a straight line. Example: <line x1="5" y1="185" x2="340" y2="185"/>

`line_element` protected member functions

1.
```
void write_attributes(std::ostream & s_out) ;
```

SVG clip path name, set & get by function clip_id.

Output group_element id and clip-path.

Classes inherited from svg_element add other references, 5.3.1, like color, fill, stroke, gradients... Example id: <g id="yMinorGrid"></g> Example class: <g class="grid_style"></g> Example URI: fill="url(#Gradient01) // local URL

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Struct m_path

boost::svg::m_path — moveto coordinates (x, y)

Synopsis

```
// In header: <boost/svg_plot/detail/svg_tag.hpp>


struct m_path : public boost::svg::path_point {
  // construct/copy/destruct
  m_path(double, double, bool = false);

  // public member functions
  void write(std::ostream &) ;
  bool relative;  // If true relative else absolute.
  double x;
  double y;
};
```

Description

8.3.2 The "moveto" commands.

`m_path` public construct/copy/destruct

1.
```
m_path(double x, double y, bool relative = false);
```

`m_path` public member functions

1.
```
void write(std::ostream & o_str) ;
```

Example: "M52.8571,180 "

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Class path_element

boost::svg::path_element — Path element holds places on a path used by move, line ...

Synopsis

```
// In header: <boost/svg_plot/detail/svg_tag.hpp>


class path_element : public boost::svg::svg_element {
public:
  // construct/copy/destruct
  path_element();
  path_element(const svg_style &, const std::string & = "",
               const std::string & = "", const std::string & = "");
  path_element(const path_element &);

  // public member functions
  path_element & C(double, double, double, double, double, double) ;
  path_element & c(double, double, double, double, double, double) ;
  std::string class_id() ;
  void class_id(const std::string &) ;
  std::string clip_id() ;
  void clip_id(const std::string &) ;
  bool fill_on() ;
  path_element & fill_on(bool) ;
  path_element & H(double) ;
  path_element & h(double) ;
  std::string id() ;
  void id(const std::string &) ;
  path_element & L(double, double) ;
  path_element & l(double, double) ;
  path_element & M(double, double) ;
  path_element & m(double, double) ;
  bool operator!=(const svg_element &) ;
  bool operator==(const svg_element &) ;
  path_element & Q(double, double, double, double) ;
  path_element & q(double, double, double, double) ;
  path_element & S(double, double, double, double) ;
  path_element & s(double, double, double, double) ;
  const svg_style & style() const;
  svg_style & style() ;
  path_element & T(double, double) ;
  path_element & t(double, double) ;
  path_element & V(double) ;
  path_element & v(double) ;
  void write(std::ostream &) ;
  path_element & Z() ;
  path_element & z() ;

  // protected member functions
  void write_attributes(std::ostream &) ;
  ptr_vector< path_point > path;

  std::string class_name_;  // SVG id name, set & get by function id.
  std::string clip_name_;  // SVG class name, set & get by function class id.
  std::string id_name_;  // Colors fill, stroke, width, get by function style.
  svg_style style_info_;
};
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Description

http://www.w3.org/TR/SVG/paths.html#PathElement 8.3.1 General information about path data. A path is defined by including a 'path' element which contains a d="(path data)" attribute, where the d attribute contains the moveto, line, curve (both cubic and quadratic Beziers), arc and closepath instructions.

`path_element` public construct/copy/destruct

1.
```
path_element();
```

2.
```
path_element(const svg_style & style_info, const std::string & id_name = "",
             const std::string & class_name = "",
             const std::string & clip_name = "");
```

3.
```
path_element(const path_element & rhs);
```

`path_element` public member functions

1.
```
path_element &
C(double x1, double y1, double x2, double y2, double x, double y) ;
```

Returns: path_element& to make chainable.

2.
```
path_element &
c(double x1, double y1, double x2, double y2, double x, double y) ;
```

Returns: path_element& to make chainable.

3.
```
std::string class_id() ;
```

Class id, non-unique identifier for an element.

4.
```
void class_id(const std::string & class_id) ;
```

Class class id, non-unique identifier for an element.

http://www.w3.org/TR/2001/REC-SVG-20010904/styling.html#ClassAttribute 6.12 Attributes common to all elements: id and xml:base Example: class="info"

5.
```
std::string clip_id() ;
```

Returns: name of a clip path, for example: g_ptr.clip_id(plot_window_clip_);

6.
```
void clip_id(const std::string & id) ;
```

Set name of a clip path, for example: g_ptr.clip_id(plot_window_clip_);

7.
```
bool fill_on() ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

8.
```
path_element & fill_on(bool on_) ;
```

Returns: path_element& to make chainable.

9.
```
path_element & H(double x) ;
```

Returns: path_element& to make chainable.

10.
```
path_element & h(double x) ;
```

Returns: path_element& to make chainable.

11.
```
std::string id() ;
```

Returns: the unique name for an element, for example id() ="plotBackground".

12.
```
void id(const std::string & id) ;
```

Provide a unique name for an element.

See http://www.w3.org/TR/SVG/struct.html#IDAttribute 5.10.1 Attributes common to all elements: id and xml:base The id and xml:base attributes are available on all SVG elements: Attribute definitions: id = "name" Standard XML attribute for assigning a unique name to an element. Refer to the "Extensible Markup Language (XML) 1.0" Recommendation [XML10]. xml:base = "<uri>" Specifies a base URI other than the base URI of the document or external entity. Refer to the "XML Base" specification [XML-BASE]. A group of elements, as well as individual objects, can be given a name using the id attribute. Named groups are needed for several purposes such as animation and re-usable objects. Example: id="plotBackground"

13.
```
path_element & L(double x, double y) ;
```

Returns: path_element& to make chainable.

14.
```
path_element & l(double x, double y) ;
```

Returns: path_element& to make chainable.

15.
```
path_element & M(double x, double y) ;
```

Returns: path_element& to make chainable.

16.
```
path_element & m(double x, double y) ;
```

Returns: path_element& to make chainable.

17.
```
bool operator!=(const svg_element & lhs) ;
```

Compare svg_elements for inequality, useful for Boost.Test.

18.
```
bool operator==(const svg_element & lhs) ;
```

Compare svg_elements, useful for Boost.Test.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

19.
```
path_element & Q(double x1, double y1, double x, double y) ;
```

Returns: path_element& to make chainable.

20.
```
path_element & q(double x1, double y1, double x, double y) ;
```

Returns: path_element& to make chainable.

21.
```
path_element & S(double x1, double y1, double x, double y) ;
```

Returns: path_element& to make chainable.

22.
```
path_element & s(double x1, double y1, double x, double y) ;
```

Returns: path_element& to make chainable.

23.
```
const svg_style & style() const;
```

Returns: reference to const svg_style to provide indirect access to colors & width via style().stroke_color(), fill_color(), width() (const version).

24.
```
svg_style & style() ;
```

Returns: reference to svg_style to provide indirect access to colors & width via style().stroke_color(), fill_color(), width()

25.
```
path_element & T(double x, double y) ;
```

Returns: path_element& to make chainable.

26.
```
path_element & t(double x, double y) ;
```

Returns: path_element& to make chainable.

27.
```
path_element & V(double y) ;
```

Returns: path_element& to make chainable.

28.
```
path_element & v(double y) ;
```

Returns: path_element& to make chainable.

29.
```
void write(std::ostream & o_str) ;
```

30.
```
path_element & Z() ;
```

Returns: path_element& to make chainable.

31.
```
path_element & z() ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Returns: path_element& to make chainable.

`path_element` protected member functions

1.
```
void write_attributes(std::ostream & s_out) ;
```

SVG clip path name, set & get by function clip_id.

Output group_element id and clip-path.

Classes inherited from svg_element add other references, 5.3.1, like color, fill, stroke, gradients... Example id: <g id="yMinorGrid"></g> Example class: <g class="grid_style"></g> Example URI: fill="url(#Gradient01) // local URL

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Struct path_point

boost::svg::path_point — Base class for m_path, z_path, q_path, h_path, v_path, c_path, s_path.

Synopsis

```
// In header: <boost/svg_plot/detail/svg_tag.hpp>


struct path_point {
  // construct/copy/destruct
  path_point(bool);
  ~path_point();

  // public member functions
  void write(std::ostream &) ;
  bool relative;  // If true relative else absolute.
};
```

Description

Paths represent the outline of a shape which can be filled, stroked, used as a clipping path, or any combination of the three.

`path_point` public construct/copy/destruct

1.
```
path_point(bool relative);
```

2.
```
~path_point();
```

`path_point` public member functions

1.
```
void write(std::ostream & rhs) ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Struct poly_path_point

boost::svg::poly_path_point — polyline or polygon point coordinates (x, y)

Synopsis

```
// In header: <boost/svg_plot/detail/svg_tag.hpp>


struct poly_path_point {
  // construct/copy/destruct
  poly_path_point();
  poly_path_point(double, double);

  // public member functions
  void write(std::ostream &) ;
  double x;
  double y;
};
```

Description

9.6 polyline & 9.7 The 'polygon' element.

`poly_path_point` public construct/copy/destruct

1.
```
poly_path_point();
```

2.
```
poly_path_point(double x, double y);
```

`poly_path_point` public member functions

1.
```
void write(std::ostream & o_str) ;
```

Output SVG XML, Example: " 250,180" Leading space is redundant for 1st after "points= ", but others are separators, and arkward to know which is 1st.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Struct polygon_element

boost::svg::polygon_element — The 'polygon' element defines a closed shape consisting of a set of connected straight line segments.

Synopsis

```
// In header: <boost/svg_plot/detail/svg_tag.hpp>


struct polygon_element : public boost::svg::svg_element {
  // construct/copy/destruct
  polygon_element(std::vector< poly_path_point > &, bool = true);
  polygon_element(double, double, double, double, double, double, double,
                  double, double, double, double, double, bool = true);
  polygon_element(double, double, double, double, double, double, double,
                  double, double, double, bool = true);
  polygon_element(double, double, double, double, double, double, double,
                  double, bool = true);
  polygon_element(double, double, double, double, double, double, bool = true);
  polygon_element(double, double, bool = true);
  polygon_element();
  polygon_element(const polygon_element &);

  // public member functions
  std::string class_id() ;
  void class_id(const std::string &) ;
  std::string clip_id() ;
  void clip_id(const std::string &) ;
  std::string id() ;
  void id(const std::string &) ;
  bool operator!=(const svg_element &) ;
  std::ostream & operator<<(std::ostream &) ;
  bool operator==(const svg_element &) ;
  polygon_element & P(double, double) ;
  const svg_style & style() const;
  svg_style & style() ;
  void write(std::ostream &) ;

  // protected member functions
  void write_attributes(std::ostream &) ;
  ptr_vector< poly_path_point > poly_points;  // All the x, y coordinate pairs, < push_backed ↵
by calls of p_path(x, y).
  bool fill;

  std::string class_name_;  // SVG id name, set & get by function id.
  std::string clip_name_;  // SVG class name, set & get by function class id.
  std::string id_name_;  // Colors fill, stroke, width, get by function style.
  svg_style style_info_;
};
```

Description

http://www.w3.org/TR/SVG/shapes.html#PolygonElement The 'polygon' element 9.9.7. A polygon is defined by including a 'path' element which contains a points="(path data)" attribute, where the d attribute contains the x, y coordinate pairs.

`polygon_element` public construct/copy/destruct

1. ```
   polygon_element(std::vector< poly_path_point > & points, bool f = true);
   ```

   Constructor from vector of path points.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/
   ```

2.
```
polygon_element(double x1, double y1, double x2, double y2, double x3,
                double y3, double x4, double y4, double x5, double y5,
                double x6, double y6, bool f = true);
```

Constructor - Six absolute (x, y) only. Used by hexagon.

3.
```
polygon_element(double x1, double y1, double x2, double y2, double x3,
                double y3, double x4, double y4, double x5, double y5,
                bool f = true);
```

Constructor - Absolute (x, y) only. Used by pentagon.

4.
```
polygon_element(double x1, double y1, double x2, double y2, double x3,
                double y3, double x4, double y4, bool f = true);
```

Constructor - Absolute (x, y) only. Used by rhombus.

5.
```
polygon_element(double x1, double y1, double x2, double y2, double x3,
                double y3, bool f = true);
```

Constructor - Absolute (x, y) only. Used by triangle.

6.
```
polygon_element(double x, double y, bool f = true);
```

Constructor - One absolute (x, y) point only. Can add more path points using member function P.

7.
```
polygon_element();
```

8.
```
polygon_element(const polygon_element & rhs);
```

`polygon_element` public member functions

1.
```
std::string class_id() ;
```

Class id, non-unique identifier for an element.

2.
```
void class_id(const std::string & class_id) ;
```

Class class id, non-unique identifier for an element.

http://www.w3.org/TR/2001/REC-SVG-20010904/styling.html#ClassAttribute 6.12 Attributes common to all elements: id and xml:base Example: class="info"

3.
```
std::string clip_id() ;
```

Returns: name of a clip path, for example: g_ptr.clip_id(plot_window_clip_);

4.
```
void clip_id(const std::string & id) ;
```

Set name of a clip path, for example: g_ptr.clip_id(plot_window_clip_);

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

5.
```
std::string id() ;
```

Returns: the unique name for an element, for example id() ="plotBackground".

6.
```
void id(const std::string & id) ;
```

Provide a unique name for an element.

See http://www.w3.org/TR/SVG/struct.html#IDAttribute 5.10.1 Attributes common to all elements: id and xml:base The id and xml:base attributes are available on all SVG elements: Attribute definitions: id = "name" Standard XML attribute for assigning a unique name to an element. Refer to the "Extensible Markup Language (XML) 1.0" Recommendation [XML10]. xml:base = "<uri>" Specifies a base URI other than the base URI of the document or external entity. Refer to the "XML Base" specification [XML-BASE]. A group of elements, as well as individual objects, can be given a name using the id attribute. Named groups are needed for several purposes such as animation and re-usable objects. Example: id="plotBackground"

7.
```
bool operator!=(const svg_element & lhs) ;
```

Compare svg_elements for inequality, useful for Boost.Test.

8.
```
std::ostream & operator<<(std::ostream & os) ;
```

Output polygon info. (May be useful for Boost.Test. using os << "(" << p.x << ", " << p.y << ")" ; Usage: polygon_element p(1, 2, 3, 4, 5, 6); my_polygon.operator << (cout); (But NOT cout << my_polygon << endl;) Outputs: (1, 2)(3, 4)(5, 6)

9.
```
bool operator==(const svg_element & lhs) ;
```

Compare svg_elements, useful for Boost.Test.

10.
```
polygon_element & P(double x, double y) ;
```

Add another point (x, y) - absolute only.

Returns: polygon_element& to make chainable.

11.
```
const svg_style & style() const;
```

Returns: reference to const svg_style to provide indirect access to colors & width via style().stroke_color(), fill_color(), width() (const version).

12.
```
svg_style & style() ;
```

Returns: reference to svg_style to provide indirect access to colors & width via style().stroke_color(), fill_color(), width()

13.
```
void write(std::ostream & o_str) ;
```

SVG XML: Example: <polygon fill="lime" stroke="blue" stroke-width="10" points="850,75 958,137.5 958,262.5 850,325 742,262.6 742,137.5" />

`polygon_element` protected member functions

1.
```
void write_attributes(std::ostream & s_out) ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

SVG clip path name, set & get by function clip_id.

Output group_element id and clip-path.

Classes inherited from svg_element add other references, 5.3.1, like color, fill, stroke, gradients... Example id: <g id="yMinorGrid"></g> Example class: <g class="grid_style"></g> Example URI: fill="url(#Gradient01) // local URL

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Class polyline_element

boost::svg::polyline_element — The 'polyline' element: defines a set of connected straight line segments.

Synopsis

```
// In header: <boost/svg_plot/detail/svg_tag.hpp>


class polyline_element : public boost::svg::svg_element {
public:
  // construct/copy/destruct
  polyline_element(std::vector< poly_path_point > &);
  polyline_element(double, double, double, double);
  polyline_element(double, double);
  polyline_element();
  polyline_element(const polyline_element &);

  // public member functions
  std::string class_id() ;
  void class_id(const std::string &) ;
  std::string clip_id() ;
  void clip_id(const std::string &) ;
  std::string id() ;
  void id(const std::string &) ;
  bool operator!=(const svg_element &) ;
  bool operator==(const svg_element &) ;
  polyline_element & P(double, double) ;
  const svg_style & style() const;
  svg_style & style() ;
  void write(std::ostream &) ;

  // protected member functions
  void write_attributes(std::ostream &) ;
  ptr_vector< poly_path_point > poly_points;  // All the (x, y) coordinate pairs,.

  std::string class_name_;  // SVG id name, set & get by function id.
  std::string clip_name_;  // SVG class name, set & get by function class id.
  std::string id_name_;  // Colors fill, stroke, width, get by function style.
  svg_style style_info_;
};
```

Description

http://www.w3.org/TR/SVG/shapes.html#PolylineElement 9.6 The polyline element: defines a set of connected straight line segments. Typically, polyline elements define open shapes. A polyline is defined by including a 'path' element which contains a points="(path data)" attribute, where the points attribute contains the x, y coordinate pairs. perform an absolute moveto operation to the first coordinate pair in the list of points for each subsequent coordinate pair, perform an absolute lineto operation to that coordinate pair. The advantage of polyline is in reducing file size, avoiding M and repeated L before x & y coordinate pairs.

`polyline_element` public construct/copy/destruct

1. ```
   polyline_element(std::vector< poly_path_point > & points);
   ```

   Constructor from vector of path points.

2. ```
   polyline_element(double x1, double y1, double x2, double y2);
   ```

   Two (x, y) path points, absolute only.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

3. ```
polyline_element(double x1, double y1);
```

One (x, y) path point, absolute only.

4. ```
polyline_element();
```

Construct an 'empty' line. Can new line path points add using polyline_element member function P.

5. ```
polyline_element(const polyline_element & rhs);
```

### `polyline_element` public member functions

1. ```
std::string class_id() ;
```

Class id, non-unique identifier for an element.

2. ```
void class_id(const std::string & class_id) ;
```

Class class id, non-unique identifier for an element.

http://www.w3.org/TR/2001/REC-SVG-20010904/styling.html#ClassAttribute 6.12 Attributes common to all elements: id and xml:base Example: class="info"

3. ```
std::string clip_id() ;
```

Returns: name of a clip path, for example: g_ptr.clip_id(plot_window_clip_);

4. ```
void clip_id(const std::string & id) ;
```

Set name of a clip path, for example: g_ptr.clip_id(plot_window_clip_);

5. ```
std::string id() ;
```

Returns: the unique name for an element, for example id() ="plotBackground".

6. ```
void id(const std::string & id) ;
```

Provide a unique name for an element.

See http://www.w3.org/TR/SVG/struct.html#IDAttribute 5.10.1 Attributes common to all elements: id and xml:base The id and xml:base attributes are available on all SVG elements: Attribute definitions: id = "name" Standard XML attribute for assigning a unique name to an element. Refer to the "Extensible Markup Language (XML) 1.0" Recommendation [XML10]. xml:base = "<uri>" Specifies a base URI other than the base URI of the document or external entity. Refer to the "XML Base" specification [XML-BASE]. A group of elements, as well as individual objects, can be given a name using the id attribute. Named groups are needed for several purposes such as animation and re-usable objects. Example: id="plotBackground"

7. ```
bool operator!=(const svg_element & lhs) ;
```

Compare svg_elements for inequality, useful for Boost.Test.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

8.
```
bool operator==(const svg_element & lhs) ;
```

Compare svg_elements, useful for Boost.Test.

9.
```
polyline_element & P(double x, double y) ;
```

Absolute (x, y) only, so Capital letter P.

Returns: polyline_element& to make chainable.

10.
```
const svg_style & style() const;
```

Returns: reference to const svg_style to provide indirect access to colors & width via style().stroke_color(), fill_color(), width() (const version).

11.
```
svg_style & style() ;
```

Returns: reference to svg_style to provide indirect access to colors & width via style().stroke_color(), fill_color(), width()

12.
```
void write(std::ostream & o_str) ;
```

Output polyline info (useful for Boost.Test). Example: <polyline points=" 100,100 200,100 300,200 400,400"/>

`polyline_element` protected member functions

1.
```
void write_attributes(std::ostream & s_out) ;
```

SVG clip path name, set & get by function clip_id.

Output group_element id and clip-path.

Classes inherited from svg_element add other references, 5.3.1, like color, fill, stroke, gradients... Example id: <g id="yMinorGrid" ></g> Example class: <g class="grid_style"></g> Example URI: fill="url(#Gradient01) // local URL

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Struct q_path

boost::svg::q_path — Draws a quadratic Bezier curve from the current point to (x,y). using (x1,y1) as the control point.

Synopsis

```
// In header: <boost/svg_plot/detail/svg_tag.hpp>


struct q_path : public boost::svg::path_point {
  // construct/copy/destruct
  q_path(double, double, double, double, bool = false);

  // public member functions
  void write(std::ostream &) ;
  bool relative;  // If true relative else absolute.
  double x;
  double x1;
  double y;
  double y1;
};
```

Description

`q_path` public construct/copy/destruct

1.
```
q_path(double x1, double y1, double x, double y, bool relative = false);
```

`q_path` public member functions

1.
```
void write(std::ostream & o_str) ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Class qurve_element

boost::svg::qurve_element — Quadratic Bezier curved line from (x1, y1) control point (x2, y2) to (x3, y3).

Synopsis

```
// In header: <boost/svg_plot/detail/svg_tag.hpp>


class qurve_element : public boost::svg::svg_element {
public:
  // construct/copy/destruct
  qurve_element(double, double, double, double, double, double,
                const svg_style &, const std::string & = "",
                const std::string & = "", const std::string & = "");
  qurve_element(double, double, double, double, double, double);

  // public member functions
  std::string class_id() ;
  void class_id(const std::string &) ;
  std::string clip_id() ;
  void clip_id(const std::string &) ;
  std::string id() ;
  void id(const std::string &) ;
  bool operator!=(const svg_element &) ;
  bool operator==(const svg_element &) ;
  const svg_style & style() const;
  svg_style & style() ;
  void write(std::ostream &) ;

  // protected member functions
  void write_attributes(std::ostream &) ;
  double x1_;  // Quadratic curved line from (x1_, y1_) control point (x2_, y2_) to (y3_, y3_).
  double x2_;  // Quadratic curved line from (x1_, y1_) control point (x2_, y2_) to (y3_, y3_).
  double x3_;  // Quadratic curved line from (x1_, y1_) control point (x2_, y2_) to (y3_, y3_).
  double y1_;  // Quadratic curved line from (x1_, y1_) control point (x2_, y2_) to (y3_, y3_).
  double y2_;  // Quadratic curved line from (x1_, y1_) control point (x2_, y2_) to (y3_, y3_).
  double y3_;  // Quadratic curved line from (x1_, y1_) control point (x2_, y2_) to (y3_, y3_).

  std::string class_name_;  // SVG id name, set & get by function id.
  std::string clip_name_;  // SVG class name, set & get by function class id.
  std::string id_name_;  // Colors fill, stroke, width, get by function style.
  svg_style style_info_;
};
```

Description

Note x2 is the Bezier control point - the curve will **not** pass thru this point.

`qurve_element` public construct/copy/destruct

1.
```
qurve_element(double x1, double y1, double x2, double y2, double x3,
              double y3, const svg_style & style_info,
              const std::string & id_name = "",
              const std::string & class_name = "",
              const std::string & clip_name = "");
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

2.
```
qurve_element(double x1, double y1, double x2, double y2, double x3,
              double y3);
```

`qurve_element` public member functions

1.
```
std::string class_id() ;
```

Class id, non-unique identifier for an element.

2.
```
void class_id(const std::string & class_id) ;
```

Class class id, non-unique identifier for an element.

http://www.w3.org/TR/2001/REC-SVG-20010904/styling.html#ClassAttribute 6.12 Attributes common to all elements: id and xml:base Example: class="info"

3.
```
std::string clip_id() ;
```

Returns: name of a clip path, for example: g_ptr.clip_id(plot_window_clip_);

4.
```
void clip_id(const std::string & id) ;
```

Set name of a clip path, for example: g_ptr.clip_id(plot_window_clip_);

5.
```
std::string id() ;
```

Returns: the unique name for an element, for example id() ="plotBackground".

6.
```
void id(const std::string & id) ;
```

Provide a unique name for an element.

See http://www.w3.org/TR/SVG/struct.html#IDAttribute 5.10.1 Attributes common to all elements: id and xml:base The id and xml:base attributes are available on all SVG elements: Attribute definitions: id = "name" Standard XML attribute for assigning a unique name to an element. Refer to the "Extensible Markup Language (XML) 1.0" Recommendation [XML10]. xml:base = "<uri>" Specifies a base URI other than the base URI of the document or external entity. Refer to the "XML Base" specification [XML-BASE]. A group of elements, as well as individual objects, can be given a name using the id attribute. Named groups are needed for several purposes such as animation and re-usable objects. Example: id="plotBackground"

7.
```
bool operator!=(const svg_element & lhs) ;
```

Compare svg_elements for inequality, useful for Boost.Test.

8.
```
bool operator==(const svg_element & lhs) ;
```

Compare svg_elements, useful for Boost.Test.

9.
```
const svg_style & style() const;
```

Returns: reference to const svg_style to provide indirect access to colors & width via style().stroke_color(), fill_color(), width() (const version).

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

10.
```
svg_style & style() ;
```

Returns: reference to svg_style to provide indirect access to colors & width via style().stroke_color(), fill_color(), width()

11.
```
void write(std::ostream & o_str) ;
```

output quadratic curved line from (x1_, y1_) control point (x2_, y2_) to (x3_, y3_)

Example:

`qurve_element` protected member functions

1.
```
void write_attributes(std::ostream & s_out) ;
```

SVG clip path name, set & get by function clip_id.

Output group_element id and clip-path.

Classes inherited from svg_element add other references, 5.3.1, like color, fill, stroke, gradients... Example id: <g id="yMinorGrid"></g> Example class: <g class="grid_style"></g> Example URI: fill="url(#Gradient01) // local URL

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Class rect_element

boost::svg::rect_element — Rectangle from top left coordinate, width and height.

Synopsis

```
// In header: <boost/svg_plot/detail/svg_tag.hpp>


class rect_element : public boost::svg::svg_element {
public:
  // construct/copy/destruct
  rect_element(double, double, double, double, const svg_style &,
               const std::string &, const std::string &, const std::string &);
  rect_element(double, double, double, double);

  // public member functions
  std::string class_id() ;
  void class_id(const std::string &) ;
  std::string clip_id() ;
  void clip_id(const std::string &) ;
  double height() const;
  std::string id() ;
  void id(const std::string &) ;
  bool operator!=(const svg_element &) ;
  bool operator!=(const rect_element &) ;
  bool operator==(const svg_element &) ;
  bool operator==(const rect_element &) ;
  const svg_style & style() const;
  svg_style & style() ;
  double width() const;
  void write(std::ostream &) ;
  double x() const;
  double y() const;

  // protected member functions
  void write_attributes(std::ostream &) ;
  double height_;  // x + width is top right.
  double width_;  // y-axis coordinate of the side of the rectangle which has the smaller y-ax↵
is coordinate value. So is top left corner of rectangle.
  double x_;
  double y_;  // x-axis coordinate of the side of the rectangle which has the smaller x-axis co↵
ordinate value.

  std::string class_name_;  // SVG id name, set & get by function id.
  std::string clip_name_;  // SVG class name, set & get by function class id.
  std::string id_name_;  // Colors fill, stroke, width, get by function style.
  svg_style style_info_;
};
```

Description

Represents a single rectangle. http://www.w3.org/TR/SVG/shapes.html#RectElement

`rect_element` public construct/copy/destruct

1.
```
rect_element(double x, double y, double w, double h,
             const svg_style & style_info, const std::string & id_name,
             const std::string & class_name, const std::string & clip_name);
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

2. ```
rect_element(double x, double y, double w, double h);
```

y + height is bottom left. x + width and y + height is bottom right.

Constructor defines all private data (no defaults).

### `rect_element` public member functions

1. ```
std::string class_id() ;
```

Class id, non-unique identifier for an element.

2. ```
void class_id(const std::string & class_id) ;
```

Class class id, non-unique identifier for an element.

http://www.w3.org/TR/2001/REC-SVG-20010904/styling.html#ClassAttribute 6.12 Attributes common to all elements: id and xml:base Example: class="info"

3. ```
std::string clip_id() ;
```

Returns: name of a clip path, for example: g_ptr.clip_id(plot_window_clip_);

4. ```
void clip_id(const std::string & id) ;
```

Set name of a clip path, for example: g_ptr.clip_id(plot_window_clip_);

5. ```
double height() const;
```

y + height is bottom left.

6. ```
std::string id() ;
```

Returns: the unique name for an element, for example id() ="plotBackground".

7. ```
void id(const std::string & id) ;
```

Provide a unique name for an element.

See http://www.w3.org/TR/SVG/struct.html#IDAttribute 5.10.1 Attributes common to all elements: id and xml:base The id and xml:base attributes are available on all SVG elements: Attribute definitions: id = "name" Standard XML attribute for assigning a unique name to an element. Refer to the "Extensible Markup Language (XML) 1.0" Recommendation [XML10]. xml:base = "<uri>" Specifies a base URI other than the base URI of the document or external entity. Refer to the "XML Base" specification [XML-BASE]. A group of elements, as well as individual objects, can be given a name using the id attribute. Named groups are needed for several purposes such as animation and re-usable objects. Example: id="plotBackground"

8. ```
bool operator!=(const svg_element & lhs) ;
```

Compare svg_elements for inequality, useful for Boost.Test.

9. ```
bool operator!=(const rect_element & lhs) ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

10.
```
bool operator==(const svg_element & lhs) ;
```

Compare svg_elements, useful for Boost.Test.

11.
```
bool operator==(const rect_element & lhs) ;
```

12.
```
const svg_style & style() const;
```

Returns: reference to const svg_style to provide indirect access to colors & width via style().stroke_color(), fill_color(), width() (const version).

13.
```
svg_style & style() ;
```

Returns: reference to svg_style to provide indirect access to colors & width via style().stroke_color(), fill_color(), width()

14.
```
double width() const;
```

x + width is top right.

15.
```
void write(std::ostream & rhs) ;
```

Output SVG XML for rectangle. For example: <rect x="0" y="0" width="500" height="350"/>

16.
```
double x() const;
```

x-axis coordinate of the side of the rectangle which has the smaller x-axis coordinate value.

17.
```
double y() const;
```

y-axis coordinate of the side of the rectangle which has the smaller y-axis coordinate value.

`rect_element` protected member functions

1.
```
void write_attributes(std::ostream & s_out) ;
```

SVG clip path name, set & get by function clip_id.

Output group_element id and clip-path.

Classes inherited from svg_element add other references, 5.3.1, like color, fill, stroke, gradients... Example id: <g id="yMinorGrid"></g> Example class: <g class="grid_style"></g> Example URI: fill="url(#Gradient01) // local URL

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Struct s_path

boost::svg::s_path — Draws a cubic Bezier curve from the current point to (x,y).

Synopsis

```
// In header: <boost/svg_plot/detail/svg_tag.hpp>


struct s_path : public boost::svg::path_point {
  // construct/copy/destruct
  s_path(double, double, double, double, bool = false);

  // public member functions
  void write(std::ostream &) ;
  bool relative;  // If true relative else absolute.
  double x;
  double x1;
  double y;
  double y1;
};
```

Description

see also t_path for a quadratic Bezier curve.

`s_path` public construct/copy/destruct

1.
```
s_path(double x1, double y1, double x, double y, bool relative = false);
```

`s_path` public member functions

1.
```
void write(std::ostream & o_str) ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Class svg_element

boost::svg::svg_element — svg_element is base class for all the leaf elements.

Synopsis

```
// In header: <boost/svg_plot/detail/svg_tag.hpp>


class svg_element {
public:
  // construct/copy/destruct
  svg_element();
  svg_element(const svg_style &, const std::string & = "",
              const std::string & = "", const std::string & = "");
  ~svg_element();

  // protected member functions
  void write_attributes(std::ostream &) ;

  // public member functions
  std::string class_id() ;
  void class_id(const std::string &) ;
  std::string clip_id() ;
  void clip_id(const std::string &) ;
  std::string id() ;
  void id(const std::string &) ;
  bool operator!=(const svg_element &) ;
  bool operator==(const svg_element &) ;
  const svg_style & style() const;
  svg_style & style() ;
  void write(std::ostream &) ;
  std::string class_name_;  // SVG id name, set & get by function id.
  std::string clip_name_;  // SVG class name, set & get by function class id.
  std::string id_name_;  // Colors fill, stroke, width, get by function style.
  svg_style style_info_;
};
```

Description

'g' element is a container element, <g ... /> </g> for grouping together related graphics elements, for example: <g stroke="rgb(255,0,0)" <rect x="0" y="0" width="500" height="600"/> </g>

rect_element, circle_element, line_element, text_element, polygon_element, polyline_element, path_element, clip_path_element, g_element.

g_element ('g' element is a container element for grouping together related graphics elements). See http://www.w3.org/TR/SVG/struct.html#NewDocument 5.2.1 Overview.

`svg_element` public construct/copy/destruct

1.
```
svg_element();
```

2.
```
svg_element(const svg_style & style_info, const std::string & id_name = "",
            const std::string & class_name = "",
            const std::string & clip_name = "");
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

3.
```
~svg_element();
```

`svg_element` protected member functions

1.
```
void write_attributes(std::ostream & s_out) ;
```

SVG clip path name, set & get by function clip_id.

Output group_element id and clip-path.

Classes inherited from svg_element add other references, 5.3.1, like color, fill, stroke, gradients... Example id: <g id="yMinorGrid"></g> Example class: <g class="grid_style"></g> Example URI: fill="url(#Gradient01) // local URL

`svg_element` public member functions

1.
```
std::string class_id() ;
```

Class id, non-unique identifier for an element.

2.
```
void class_id(const std::string & class_id) ;
```

Class class id, non-unique identifier for an element.

http://www.w3.org/TR/2001/REC-SVG-20010904/styling.html#ClassAttribute 6.12 Attributes common to all elements: id and xml:base Example: class="info"

3.
```
std::string clip_id() ;
```

Returns: name of a clip path, for example: g_ptr.clip_id(plot_window_clip_);

4.
```
void clip_id(const std::string & id) ;
```

Set name of a clip path, for example: g_ptr.clip_id(plot_window_clip_);

5.
```
std::string id() ;
```

Returns: the unique name for an element, for example id() ="plotBackground".

6.
```
void id(const std::string & id) ;
```

Provide a unique name for an element.

See http://www.w3.org/TR/SVG/struct.html#IDAttribute 5.10.1 Attributes common to all elements: id and xml:base The id and xml:base attributes are available on all SVG elements: Attribute definitions: id = "name" Standard XML attribute for assigning a unique name to an element. Refer to the "Extensible Markup Language (XML) 1.0" Recommendation [XML10]. xml:base = "<uri>" Specifies a base URI other than the base URI of the document or external entity. Refer to the "XML Base" specification [XML-BASE]. A group of elements, as well as individual objects, can be given a name using the id attribute. Named groups are needed for several purposes such as animation and re-usable objects. Example: id="plotBackground"

7.
```
bool operator!=(const svg_element & lhs) ;
```

Compare svg_elements for inequality, useful for Boost.Test.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

8.
```
bool operator==(const svg_element & lhs) ;
```

Compare svg_elements, useful for Boost.Test.

9.
```
const svg_style & style() const;
```

Returns: reference to const svg_style to provide indirect access to colors & width via style().stroke_color(), fill_color(), width() (const version).

10.
```
svg_style & style() ;
```

Returns: reference to svg_style to provide indirect access to colors & width via style().stroke_color(), fill_color(), width()

11.
```
void write(std::ostream & rhs) ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Struct t_path

boost::svg::t_path — Draws a quadratic Bezier curve from the current point to (x,y).

Synopsis

```
// In header: <boost/svg_plot/detail/svg_tag.hpp>


struct t_path : public boost::svg::path_point {
  // construct/copy/destruct
  t_path(double, double, bool = false);

  // public member functions
  void write(std::ostream &) ;
  bool relative;  // If true relative else absolute.
  double x;
  double y;
};
```

Description

see also s_path for a cubic Bezier curve.

`t_path` public construct/copy/destruct

1.
```
t_path(double x, double y, bool relative = false);
```

`t_path` public member functions

1.
```
void write(std::ostream & o_str) ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Class text_element

boost::svg::text_element — Holds text with position, size, font, (& styles) & orientation.

Synopsis

```
// In header: <boost/svg_plot/detail/svg_tag.hpp>


class text_element : public boost::svg::svg_element {
public:
  // construct/copy/destruct
  text_element(const text_element &);
  text_element(double = 0., double = 0., const std::string = "",
               text_style = no_style, align_style = left_align,
               rotate_style = horizontal);
  text_element& operator=(const text_element &);

  // private member functions
  void generate_text(std::ostream &) ;

  // public member functions
  align_style alignment() ;
  text_element & alignment(align_style) ;
  std::string class_id() ;
  void class_id(const std::string &) ;
  std::string clip_id() ;
  void clip_id(const std::string &) ;
  std::string id() ;
  void id(const std::string &) ;
  bool operator!=(const svg_element &) ;
  bool operator==(const svg_element &) ;
  rotate_style rotation() const;
  text_element & rotation(rotate_style) ;
  const svg_style & style() const;
  svg_style & style() ;
  std::string text() ;
  void text(const std::string &) ;
  text_element & textstyle(text_style &) ;
  const text_style & textstyle() const;
  text_style & textstyle() ;
  tspan_element & tspan(const std::string &, const text_style &) ;
  tspan_element & tspan(const std::string &) ;
  void write(std::ostream &) ;
  double x() const;
  text_element & x(double) ;
  double y() const;
  text_element & y(double) ;

  // protected member functions
  void write_attributes(std::ostream &) ;
  align_style align_;  // left_align, right_align, center_align
  ptr_vector< text_parent > data_;  // Stores all of the containing data.
  rotate_style rotate_;  // horizontal, upward, downward, upsidedown
  text_style style_;  // font variants.
  double x_;  // Left edge.
  double y_;  // Bottom of roman capital character.
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
  std::string class_name_;  // SVG id name, set & get by function id.
  std::string clip_name_;  // SVG class name, set & get by function class id.
  std::string id_name_;  // Colors fill, stroke, width, get by function style.
  svg_style style_info_;
};
```

Description

Not necessarily shown correctly (or nicely) by all browsers, alas. SVG Coordinates of 1st character EM box, see http://www.w3.org/TR/SVG/text.html#TextElement 10.2 So any text with y coordinate = 0 shows only any roman lower case descenders!\n\n (Text may contain embedded xml Unicode characters for Greek, math etc, for example: Ω). int size; // " font-size = 12" http://www.w3.org/TR/SVG/text.html#CharactersAndGlyphs std::string font; // font-family: "Arial" | "Times New Roman" | "Verdana" | "Lucida Sans Unicode" "sans", "serif", "times" http://www.w3.org/TR/SVG/text.html#FontFamilyProperty 10.10 Font selection properties std::string style_; // font-style: normal | bold | italic | oblique std::string weight; // font-weight: normal | bold | bolder | lighter | 100 | 200 .. 900 std::string stretch; // font-stretch: normal | wider | narrower ... std::string decoration; // // "underline" | "overline" | "line-through" Example: <text x="250" y="219.5" text-anchor="middle" font-family="verdana" font-size="12">0 </text>

`text_element` public construct/copy/destruct

1.
```
text_element(const text_element & rhs);
```

2.
```
text_element(double x = 0., double y = 0., const std::string text = "",
             text_style ts = no_style, align_style align = left_align,
             rotate_style rotate = horizontal);
```

text_element Default Constructor, defines defaults for all private members.

3.
```
text_element& operator=(const text_element & rhs);
```

to make chainable.

`text_element` private member functions

1.
```
void generate_text(std::ostream & os) ;
```

`text_element` public member functions

1.
```
align_style alignment() ;
```

left_align, right_align, center_align

2.
```
text_element & alignment(align_style a) ;
```

left_align, right_align, center_align

Returns: text_element& to make chainable.

3.
```
std::string class_id() ;
```

Class id, non-unique identifier for an element.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

4.
```
void class_id(const std::string & class_id) ;
```

Class class id, non-unique identifier for an element.

http://www.w3.org/TR/2001/REC-SVG-20010904/styling.html#ClassAttribute 6.12 Attributes common to all elements: id and xml:base Example: class="info"

5.
```
std::string clip_id() ;
```

Returns: name of a clip path, for example: g_ptr.clip_id(plot_window_clip_);

6.
```
void clip_id(const std::string & id) ;
```

Set name of a clip path, for example: g_ptr.clip_id(plot_window_clip_);

7.
```
std::string id() ;
```

Returns: the unique name for an element, for example id() ="plotBackground".

8.
```
void id(const std::string & id) ;
```

Provide a unique name for an element.

See http://www.w3.org/TR/SVG/struct.html#IDAttribute 5.10.1 Attributes common to all elements: id and xml:base The id and xml:base attributes are available on all SVG elements: Attribute definitions: id = "name" Standard XML attribute for assigning a unique name to an element. Refer to the "Extensible Markup Language (XML) 1.0" Recommendation [XML10]. xml:base = "<uri>" Specifies a base URI other than the base URI of the document or external entity. Refer to the "XML Base" specification [XML-BASE]. A group of elements, as well as individual objects, can be given a name using the id attribute. Named groups are needed for several purposes such as animation and re-usable objects. Example: id="plotBackground"

9.
```
bool operator!=(const svg_element & lhs) ;
```

Compare svg_elements for inequality, useful for Boost.Test.

10.
```
bool operator==(const svg_element & lhs) ;
```

Compare svg_elements, useful for Boost.Test.

11.
```
rotate_style rotation() const;
```

12.
```
text_element & rotation(rotate_style rot) ;
```

Degrees: horizontal = 0, upward = -90, downward, upsidedown Generates: transform = "rotate(-45 100 100)"

Returns: text_element& to make chainable.

13.
```
const svg_style & style() const;
```

Returns: reference to const svg_style to provide indirect access to colors & width via style().stroke_color(), fill_color(), width() (const version).

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

14.
```
svg_style & style() ;
```

Returns: reference to svg_style to provide indirect access to colors & width via style().stroke_color(), fill_color(), width()

15.
```
std::string text() ;
```

16.
```
void text(const std::string & t) ;
```

Get tspan text string to write.

17.
```
text_element & textstyle(text_style & ts) ;
```

Set text style for font size, family, decoration ...

Returns: text_element& to make chainable.

18.
```
const text_style & textstyle() const;
```

Get text style for font size, family, decoration ...

19.
```
text_style & textstyle() ;
```

Get text style for font size, family, decoration ...

20.
```
tspan_element & tspan(const std::string & t, const text_style & style) ;
```

Add text span element (with specified text style).

21.
```
tspan_element & tspan(const std::string & t) ;
```

Add text span element.

22.
```
void write(std::ostream & os) ;
```

Output text_element, style & attributes to stream.

23.
```
double x() const;
```

x coordinate of text to write.

24.
```
text_element & x(double x) ;
```

x coordinate of text to write.

Returns: text_element& to make chainable.

25.
```
double y() const;
```

y coordinate of text to write.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

26.
```
text_element & y(double y) ;
```

y coordinate of text to write.

Returns: text_element& to make chainable.

`text_element` protected member functions

1.
```
void write_attributes(std::ostream & s_out) ;
```

SVG clip path name, set & get by function clip_id.

Output group_element id and clip-path.

Classes inherited from svg_element add other references, 5.3.1, like color, fill, stroke, gradients... Example id: <g id="yMinorGrid"></g> Example class: <g class="grid_style"></g> Example URI: fill="url(#Gradient01) // local URL

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Class text_element_text

boost::svg::text_element_text — text (not tspan) element to be stored in text_parent.

Synopsis

```
// In header: <boost/svg_plot/detail/svg_tag.hpp>


class text_element_text : public boost::svg::text_parent {
public:
  // construct/copy/destruct
  text_element_text(const text_element_text &);
  text_element_text(const std::string &);

  // public member functions
  void write(std::ostream &) ;
  std::string text_;
};
```

Description

See 10.4 text element http://www.w3.org/TR/SVG/text.html#TextElement

`text_element_text` public construct/copy/destruct

1.
```
text_element_text(const text_element_text & rhs);
```

2.
```
text_element_text(const std::string & text);
```

`text_element_text` public member functions

1.
```
void write(std::ostream & o_str) ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Class text_parent

boost::svg::text_parent — An ancestor to both tspan and strings for the text_element class.

Synopsis

```
// In header: <boost/svg_plot/detail/svg_tag.hpp>


class text_parent {
public:
  // construct/copy/destruct
  text_parent(const text_parent &);
  text_parent(const std::string &);

  // public member functions
  void write(std::ostream &) ;
  std::string text_;
};
```

Description

This allows an array of both types to be stored in text_element.

`text_parent` public construct/copy/destruct

1.
```
text_parent(const text_parent & rhs);
```

2.
```
text_parent(const std::string & text);
```

`text_parent` public member functions

1.
```
void write(std::ostream &) ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Class tspan_element

boost::svg::tspan_element — tspan (not text) to be stored in text_parent.

Synopsis

```
// In header: <boost/svg_plot/detail/svg_tag.hpp>


class tspan_element :
  public boost::svg::text_parent, public boost::svg::svg_element
{
public:
  // construct/copy/destruct
  tspan_element(const tspan_element &);
  tspan_element(const std::string &, const text_style & = no_style);

  // public member functions
  std::string class_id() ;
  void class_id(const std::string &) ;
  std::string clip_id() ;
  void clip_id(const std::string &) ;
  double dx() ;
  tspan_element & dx(double) ;
  double dy() ;
  tspan_element & dy(double) ;
  svg_color fill_color() ;
  tspan_element & fill_color(const svg_color &) ;
  bool fill_on() ;
  const std::string & font_family() ;
  tspan_element & font_family(const std::string &) ;
  unsigned int font_size() ;
  tspan_element & font_size(unsigned int) ;
  const std::string & font_style() const;
  const std::string & font_style() ;
  tspan_element & font_style(const std::string &) ;
  const std::string & font_weight() const;
  tspan_element & font_weight(const std::string &) ;
  std::string id() ;
  void id(const std::string &) ;
  bool operator!=(const svg_element &) ;
  bool operator==(const svg_element &) ;
  int rotation() ;
  tspan_element & rotation(int) ;
  const svg_style & style() const;
  svg_style & style() ;
  std::string text() ;
  tspan_element & text(const std::string &) ;
  double text_length() ;
  tspan_element & text_length(double) ;
  const text_style & textstyle() const;
  const text_style & textstyle() ;
  tspan_element & textstyle(const text_style &) ;
  bool use_style() ;
  void write(std::ostream &) ;
  double x() ;
  tspan_element & x(double) ;
  double y() ;
  tspan_element & y(double) ;

  // protected member functions
  void write_attributes(std::ostream &) ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
  double dx_;  // Relative X position of a 1st single character of text.
  double dy_;  // Relative Y position of a 1st single character of text.
  int rotate_;  // Rotation of a 1st single character of text.
  text_style style_;  // font variants.
  double text_length_;  // Allows the author to provide exact alignment. dx_, dy_, and rotate_ ↵
can all be omitted, usually meaning no shift or rotation, but see ht↵
tp://www.w3.org/TR/SVG/text.html#TSpanElement for ancestor rules. but x_, y_, and text_length ↵
need a flag.
  bool use_style_;
  bool use_text_length_;  // > If true then use Y absolute position.
  bool use_x_;
  bool use_y_;  // > If true then use X absolute position.
  double x_;  // Absolute X position.
  double y_;  // Absolute Y position.

  std::string class_name_;  // SVG id name, set & get by function id.
  std::string clip_name_;  // SVG class name, set & get by function class id.
  std::string id_name_;  // Colors fill, stroke, width, get by function style.
  svg_style style_info_;
  std::string text_;
};
```

Description

See 10.5 tspan element http://www.w3.org/TR/SVG/text.html#TSpanElement

`tspan_element` public construct/copy/destruct

1.
```
tspan_element(const tspan_element & rhs);
```

2.
```
tspan_element(const std::string & text, const text_style & style = no_style);
```

`tspan_element` public member functions

1.
```
std::string class_id() ;
```

Class id, non-unique identifier for an element.

2.
```
void class_id(const std::string & class_id) ;
```

Class class id, non-unique identifier for an element.

http://www.w3.org/TR/2001/REC-SVG-20010904/styling.html#ClassAttribute 6.12 Attributes common to all elements: id and xml:base Example: class="info"

3.
```
std::string clip_id() ;
```

Returns: name of a clip path, for example: g_ptr.clip_id(plot_window_clip_);

4.
```
void clip_id(const std::string & id) ;
```

Set name of a clip path, for example: g_ptr.clip_id(plot_window_clip_);

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

5. ```
double dx() ;
```

Get relative X position for tspan element.

6. ```
tspan_element & dx(double dx) ;
```

Returns: tspan_element& to make chainable.

7. ```
double dy() ;
```

Get relative Y position for tspan element.

8. ```
tspan_element & dy(double dy) ;
```

Returns: tspan_element& to make chainable.

9. ```
svg_color fill_color() ;
```

Get the fill color for tspan element .

10. ```
tspan_element & fill_color(const svg_color & color) ;
```

Set fill color for a tspan element.

Returns: tspan_element& to make chainable.

11. ```
bool fill_on() ;
```

Get true if to use fill color for tspan element .

12. ```
const std::string & font_family() ;
```

Get the font family for tspan element (from its text_style).

13. ```
tspan_element & font_family(const std::string & family) ;
```

Returns: tspan_element& to make chainable.

14. ```
unsigned int font_size() ;
```

Get the font size for tspan element (from its text_style).

15. ```
tspan_element & font_size(unsigned int size) ;
```

Returns: tspan_element& to make chainable.

16. ```
const std::string & font_style() const;
```

Get the font style for tspan element (from its text_style). const version.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

17.
```
const std::string & font_style() ;
```

Get the font style for tspan element (from its text_style).

18.
```
tspan_element & font_style(const std::string & style) ;
```

font-style: normal | bold | italic | oblique Examples: "italic" http://www.croczilla.com/~alex/conformance_suite/svg/text-fonts-02-t.svg

Returns: tspan_element& to make chainable.

19.
```
const std::string & font_weight() const;
```

Get the font weight for tspan element (from its text_style).

20.
```
tspan_element & font_weight(const std::string & w) ;
```

svg font-weight: normal | bold | bolder | lighter | 100 | 200 .. 900 Examples: "bold", "normal" http://www.croczilla.com/~alex/conformance_suite/svg/text-fonts-02-t.svg tests conformance. Only two weights are supported by Firefox, Opera, Inkscape.

Returns: tspan_element& to make chainable.

21.
```
std::string id() ;
```

Returns: the unique name for an element, for example id() ="plotBackground".

22.
```
void id(const std::string & id) ;
```

Provide a unique name for an element.

See http://www.w3.org/TR/SVG/struct.html#IDAttribute 5.10.1 Attributes common to all elements: id and xml:base The id and xml:base attributes are available on all SVG elements: Attribute definitions: id = "name" Standard XML attribute for assigning a unique name to an element. Refer to the "Extensible Markup Language (XML) 1.0" Recommendation [XML10]. xml:base = "<uri>" Specifies a base URI other than the base URI of the document or external entity. Refer to the "XML Base" specification [XML-BASE]. A group of elements, as well as individual objects, can be given a name using the id attribute. Named groups are needed for several purposes such as animation and re-usable objects. Example: id="plotBackground"

23.
```
bool operator!=(const svg_element & lhs) ;
```

Compare svg_elements for inequality, useful for Boost.Test.

24.
```
bool operator==(const svg_element & lhs) ;
```

Compare svg_elements, useful for Boost.Test.

25.
```
int rotation() ;
```

Get rotation for the next character for tspan element.

26.
```
tspan_element & rotation(int rotation) ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

< Note implementation so far only rotates the 1st character in string. < text_element rotation rotates the whole text string, so it *much* more useful.

Returns: tspan_element& to make chainable.

27.
```
const svg_style & style() const;
```

Returns: reference to const svg_style to provide indirect access to colors & width via style().stroke_color(), fill_color(), width() (const version).

28.
```
svg_style & style() ;
```

Returns: reference to svg_style to provide indirect access to colors & width via style().stroke_color(), fill_color(), width()

29.
```
std::string text() ;
```

Get text from a tspan element.

30.
```
tspan_element & text(const std::string & text) ;
```

Returns: tspan_element& to make chainable.

31.
```
double text_length() ;
```

Get user estimated length for a text string.

32.
```
tspan_element & text_length(double text_length) ;
```

Set user estimate of text length (see http://www.w3.org/TR/SVG/text.html#TSpanElement TSPAN SVG Specification).

Returns: tspan_element& to make chainable.

33.
```
const text_style & textstyle() const;
```

34.
```
const text_style & textstyle() ;
```

35.
```
tspan_element & textstyle(const text_style & style) ;
```

Set text style for a tspan element.

Returns: tspan_element& to make chainable.

36.
```
bool use_style() ;
```

Get true if to use the estimated text string length.

37.
```
void write(std::ostream & os) ;
```

Output SVG XML for tspan_element

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

38.
```
double x() ;
```

Get absolute X position for tspan element.

39.
```
tspan_element & x(double x) ;
```

Returns: tspan_element& to make chainable.

40.
```
double y() ;
```

Get absolute Y position for tspan element.

41.
```
tspan_element & y(double y) ;
```

Returns: tspan_element& to make chainable.

`tspan_element` protected member functions

1.
```
void write_attributes(std::ostream & s_out) ;
```

SVG clip path name, set & get by function clip_id.

Output group_element id and clip-path.

Classes inherited from svg_element add other references, 5.3.1, like color, fill, stroke, gradients... Example id: <g id="yMinorGrid"></g> Example class: <g class="grid_style"></g> Example URI: fill="url(#Gradient01) // local URL

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Struct v_path

boost::svg::v_path — Draws a vertical line from the current point (cpx, cpy) to (cpx, y). No x coordinate needed - use current point x.

Synopsis

```
// In header: <boost/svg_plot/detail/svg_tag.hpp>


struct v_path : public boost::svg::path_point {
  // construct/copy/destruct
  v_path(double, bool = false);

  // public member functions
  void write(std::ostream &) ;
  bool relative;  // If true relative else absolute.
  double y;
};
```

Description

`v_path` public construct/copy/destruct

1.
```
v_path(double y, bool relative = false);
```

`v_path` public member functions

1.
```
void write(std::ostream & o_str) ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Struct z_path

boost::svg::z_path — Close current path.

Synopsis

```
// In header: <boost/svg_plot/detail/svg_tag.hpp>


struct z_path : public boost::svg::path_point {
  // construct/copy/destruct
  z_path();

  // public member functions
  void write(std::ostream &) ;
  bool relative;  // If true relative else absolute.
};
```

Description

http://www.w3.org/TR/SVG/paths.html#PathElement 8.3.1 General information about path data. Close the current subpath by drawing a straight line from the current point to current subpath's initial point.

`z_path` public construct/copy/destruct

1.
```
z_path();
```

`z_path` public member functions

1.
```
void write(std::ostream & o_str) ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Function operator!=

boost::svg::operator!=

Synopsis

```
// In header: <boost/svg_plot/detail/svg_tag.hpp>


bool operator!=(const rect_element & lhs, const rect_element & rhs);
```

Description

Compare inequality of two SVG rect_elements.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Function operator<<

boost::svg::operator<<

Synopsis

```
// In header: <boost/svg_plot/detail/svg_tag.hpp>


std::ostream & operator<<(std::ostream & os, polyline_element & p);
```

Description

Output polyline info (useful for Boost.Test). Example: <polyline points=" 100,100 200,100 300,200 400,400"/> ptr_vector<poly_path_point> poly_points; // All the x, y coordinate pairs.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Function operator<<

boost::svg::operator<<

Synopsis

```
// In header: <boost/svg_plot/detail/svg_tag.hpp>


std::ostream & operator<<(std::ostream & os, polygon_element & p);
```

Description

Output poly_path_ points (May be useful for Boost.Test). ptr_vector<poly_path_point> poly_points; All the x, y coordinate pairs, Usage: polygon_element p(1, 2, 3, 4, 5, 6); cout << p << endl; Outputs: (1, 2)(3, 4)(5, 6)

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Function operator<<

boost::svg::operator<<

Synopsis

```
// In header: <boost/svg_plot/detail/svg_tag.hpp>


std::ostream & operator<<(std::ostream & os, const poly_path_point & p);
```

Description

Output may be useful for Boost.Test. Usage: poly_path_point p0(100, 200); cout << p0 << endl; Outputs: (100, 200)

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Function operator<<

boost::svg::operator<<

Synopsis

```
// In header: <boost/svg_plot/detail/svg_tag.hpp>


std::ostream & operator<<(std::ostream & os, text_element & t);
```

Description

Outputs: text & style (useful for diagnosis). Usage: text_element t(20, 30, "sometest", left_align, horizontal); cout << t << endl;

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Function operator<<

boost::svg::operator<<

Synopsis

```
// In header: <boost/svg_plot/detail/svg_tag.hpp>


std::ostream & operator<<(std::ostream & os, const rect_element & r);
```

Description

Example: rect_element r(20, 20, 50, 50); cout << r << endl; Outputs: rect(20, 20, 50, 50)

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Function operator==

boost::svg::operator==

Synopsis

```
// In header: <boost/svg_plot/detail/svg_tag.hpp>


bool operator==(const rect_element & lhs, const rect_element & rhs);
```

Description

Compare equality of two SVG rect_elements.

Header <boost/svg_plot/quantile.hpp>

Estimate p th quantile of data.

Paul A. Bristow
Estimate p th quantile of data using one of 5 definitions. Default is the recommendation of Hyndman and Fan = definition #8.

Hyndman and Fan recommend their definition 8 (Maple's default definition), which gives quartiles between those reported by Minitab and Excel. This approach is approximately median unbiased for continuous distributions.

Hyndman and Fan, 1996, "Sample Quantiles in Statistical Packages", The American Statistician 50(4):361-365,1996 http://www.pcreview.co.uk/forums/thread-3494699.php // Excel - Interquartile Range Miscalculation

The interquartile range is calculated using the 1st & 3rd sample quartiles, but there are various ways to calculate those quartiles. Excel, S-Plus, etc use H&F definition 7, which returns SMALL(data,i) as quantile(data,(i-1)/(n-1)) and interpolates in between. For a continuous distribution, this will tend to give too narrow an interquartile range, since there will tend to be a small fraction of the population beyond the extreme sample observations. In particular, for odd n (=2*k+1), Excel calculates the 1st (3rd) quartile as the median of the lower (upper) "half" of the sample including the sample median (k+1 observations).

Minitab, etc use H&F definition 6, which calculates the 1st (3rd) quartile as the median of the lower (upper) "half" of the sample. This "half" sample excludes the sample median (k observations) for odd n (=2*k+1). This will tend to be a better estimate for the population quartiles, but will tend to give quartile estimates that are a bit too far from the center of the whole sample (too wide an interquartile range).

```
namespace boost {
  namespace svg {
    double median(vector< double > &);
    double quantile(vector< double > &, double, int = 8);
  }
}
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Function median

boost::svg::median

Synopsis

```
// In header: <boost/svg_plot/quantile.hpp>


double median(vector< double > & data);
```

Description

Median of data - assumed pre-sorted from min to max.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Function quantile

boost::svg::quantile

Synopsis

```
// In header: <boost/svg_plot/quantile.hpp>


double quantile(vector< double > & data, double p, int HF_definition = 8);
```

Description

Estimate quantile from data.

Parameters:	`HF_definition`	algorithm to use for the estimation.
	`data`	population for which to estimate quantile (percentile). Data must be ordered minimum to maximum.
	`p`	fraction of population, for example, p = 0.25 for 1st quartile. (Usually p = 0.25, or p = 0.75 for boxplots).
Returns:	quantile estimated quantile from the population.	

Header <boost/svg_plot/show_1d_settings.hpp>

Shows settings and options for 1-D Plot.

Jacob Voytko & Paul A. Bristow
Outputs a full list of all settings and options for 1-D Plot to std::cout. This is useful for diagnosing why your plot doesn't look as you hoped!

This creates about 100 lines of output, so should be used sparingly!

Paul A. Bristow

Mar 2009

show_2d_settings.cpp for 2-D plot.

```
namespace boost {
  namespace svg {
    const char * fmtFlagWords;    // Strings describing each bit in std::ios_base::fmtflags.

    // Outputs strings to show horizontal orientation: left, right or none.
    const std::string l_or_r(int i);
    void outFmtFlags(std::ios_base::fmtflags, std::ostream &, const char *);
    void show_1d_plot_settings(svg_1d_plot &);

    // Outputs strings to show vertical orientation: top, bottom, or none.
    const std::string t_or_b(int i);
  }
}
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Global fmtFlagWords

boost::svg::fmtFlagWords — Strings describing each bit in std::ios_base::fmtflags.

Synopsis

```
// In header: <boost/svg_plot/show_1d_settings.hpp>

const char * fmtFlagWords;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Function outFmtFlags

boost::svg::outFmtFlags

Synopsis

```
// In header: <boost/svg_plot/show_1d_settings.hpp>


void outFmtFlags(std::ios_base::fmtflags fmtFlags, std::ostream & os,
                 const char * term);
```

Description

Output strings describing all bits in std::ios_base::fmtflags.

Usage: outFmtFlags(); for example, outputs to cerr: "FormatFlags: skipws showbase right dec" Default parameter values are: void outFmtFlags(fmtflags fmtFlags = cout.flags(), ostream& os = cerr, const char* term = ".\n");

Output strings describing format flags of each bit in std::ios_base::fmtflags.

Usage: outFmtFlags(flags, cerr); For example, logs to cerr "FormatFlags: skipws showbase right dec"

Output strings describing format flags of each bit in std::ios_base::fmtflags.

Usage: outFmtFlags(flags, cerr); For example, logs to cerr "FormatFlags: skipws showbase right dec"

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Function show_1d_plot_settings

boost::svg::show_1d_plot_settings

Synopsis

```
// In header: <boost/svg_plot/show_1d_settings.hpp>


void show_1d_plot_settings(svg_1d_plot & plot);
```

Description

Diagnostic display to std::cout of all of a 1D plot's settings.

Outputs to cout, a long list of about hundred of plot parameter settings, invaluable if the plot does not look as expected.

This creates about 100 lines of output, so should be used sparingly!
Diagnostic display to std::cout of all of a 1D plot's settings.

Outputs to cout, a long list of about hundred of plot parameter settings, invaluable if the plot does not look as expected.

This creates about 100 lines of output, so should be used sparingly!

Header <boost/svg_plot/show_2d_settings.hpp>

Shows settings and options for 2D Plot.

Jacob Voytko & Paul A. Bristow
Outputs a full list of all settings and options for 2D Plot to std::cout. This is useful for diagnosing why your plot doesn't look as you hoped! (See also show_1d_settings.cpp for 1D plot).

This creates about 100 lines of output, so should be used sparingly!

```
namespace boost {
  namespace svg {
    void show_2d_plot_settings(svg_2d_plot &);
  }
}
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Function show_2d_plot_settings

boost::svg::show_2d_plot_settings

Synopsis

```
// In header: <boost/svg_plot/show_2d_settings.hpp>


void show_2d_plot_settings(svg_2d_plot & plot);
```

Description

Diagnostic display of all of a 2D plot's settings. Outputs to cout, a long list of about hundred of plot parameter settings, invaluable if the plot does not look as expected.

This creates about 100 lines of output, so should be used sparingly!
Diagnostic display of all of a 2D plot's settings. Outputs to cout, a long list of about hundred of plot parameter settings, invaluable if the plot does not look as expected.

This creates about 100 lines of output, so should be used sparingly!

Header <boost/svg_plot/stylesheet.hpp>

Implement Cascading Style Sheets for SVG plots (but NOT IMPLEMENTED yet).

Get some inscrutable errors from this and so commented out of other modules.

Jacob Voytko

2007

```
namespace boost {
  namespace svg {
    struct css_parse;

    // Validate a style sheet from file.
    bool validate_stylesheet(const std::string & file);
    bool validate_stylesheet(std::ifstream &);
  }
}
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Struct css_parse

boost::svg::css_parse — CSS parser.

Synopsis

```
// In header: <boost/svg_plot/stylesheet.hpp>


struct css_parse {

  // CSS parser definition.   template<typename ScannerT >
  struct definition {
    // construct/copy/destruct
    definition(css_parse const &);

    // public member functions
    ::boost::spirit::rule< ScannerT > const & start() const;
    ::boost::spirit::rule< ScannerT > class_expression;
    ::boost::spirit::rule< ScannerT > expression;
    ::boost::spirit::rule< ScannerT > id_expression;
    ::boost::spirit::rule< ScannerT > identifier;
    ::boost::spirit::rule< ScannerT > normal_expression;
    ::boost::spirit::rule< ScannerT > statement;
    ::boost::spirit::rule< ScannerT > wspace;
  };
};
```

Description

not used and need replementing to useW3C CSS 2.1 Standard

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Struct template definition

boost::svg::css_parse::definition — CSS parser definition.

Synopsis

```
// In header: <boost/svg_plot/stylesheet.hpp>



// CSS parser definition. template<typename ScannerT >
struct definition {
  // construct/copy/destruct
  definition(css_parse const &);

  // public member functions
  ::boost::spirit::rule< ScannerT > const & start() const;
  ::boost::spirit::rule< ScannerT > class_expression;
  ::boost::spirit::rule< ScannerT > expression;
  ::boost::spirit::rule< ScannerT > id_expression;
  ::boost::spirit::rule< ScannerT > identifier;
  ::boost::spirit::rule< ScannerT > normal_expression;
  ::boost::spirit::rule< ScannerT > statement;
  ::boost::spirit::rule< ScannerT > wspace;
};
```

Description

not used and need replementing to useW3C CSS 2.1 Standard

`definition` public construct/copy/destruct

1.
```
definition(css_parse const &);
```

`definition` public member functions

1.
```
::boost::spirit::rule< ScannerT > const & start() const;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Function validate_stylesheet

boost::svg::validate_stylesheet — Validate a style sheet from istream.

Synopsis

```
// In header: <boost/svg_plot/stylesheet.hpp>


bool validate_stylesheet(std::ifstream & file);
```

Description

stream version used by file version below.

Header <boost/svg_plot/svg.hpp>

Scalable Vector Graphic (SVG) format elements.

Provides classes and methods to create the basic SVG graph elements. Graph elements, point, path, line, circle, rect and polygon, text are used by the 1D, 2D and Boxplot functions, but could also be used for generating other graphics in SVG format.

1.1

Jacob Voytko & Paul A. Bristow

Mar 2009

```
namespace boost {
  namespace math {
  }
  namespace svg {
    class svg;

    static const std::string package_info;    // Default SVG package information about this pro↵
gram that produced the SVG image (not the image itself).
    namespace boxplot {
    }
  }
}namespace svg {
}
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Class svg

boost::svg::svg — Class to output Scalable Vector Graph XML graph elements: point, path, line, circle, rect, polygon and text.

Synopsis

```
// In header: <boost/svg_plot/svg.hpp>


class svg {
public:
  // construct/copy/destruct
  svg(const svg &);
  svg();

  // private member functions
  void write_css(std::ostream &) ;
  void write_document(std::ostream &) ;
  void write_header(std::ostream &) ;

  // public member functions
  g_element & add_g_element() ;
  const std::string & attribution() ;
  const std::string & author() ;
  void author(const std::string) ;
  bool boost_license_on() ;
  void boost_license_on(bool) ;
  circle_element & circle(double, double, unsigned int = 5) ;
  clip_path_element & clip_path(const rect_element &, const std::string &) ;
  const std::string & commercialuse() ;
  int coord_precision() ;
  void coord_precision(int) ;
  const std::string copyright_date() ;
  void copyright_date(const std::string) ;
  const std::string copyright_holder() ;
  void copyright_holder(const std::string) ;
  const std::string & description() ;
  void description(const std::string) ;
  const std::string & distribution() ;
  unsigned int document_size() ;
  const std::string document_title() ;
  void document_title(const std::string) ;
  ellipse_element & ellipse(double, double, double, double) ;
  g_element & g(int) ;
  polygon_element &
  hexagon(double, double, double, double, double, double, double, double,
          double, double, double, double, bool = true) ;
  const std::string image_filename() ;
  void image_filename(const std::string) ;
  void license(const std::string = "permits", const std::string = "permits",
               const std::string = "requires", const std::string = "permits",
               const std::string = "permits") ;
  bool license_on() ;
  void license_on(bool) ;
  line_element & line(double, double, double, double) ;
  path_element & path() ;
  polygon_element &
  pentagon(double, double, double, double, double, double, double, double,
           double, double, bool = true) ;
  polygon_element & polygon(std::vector< poly_path_point > &, bool = true) ;
  polygon_element & polygon(double, double, bool = true) ;
  polyline_element & polyline(std::vector< poly_path_point > &) ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
  polyline_element & polyline(double, double, double, double) ;
  polyline_element & polyline(double, double) ;
  rect_element & rect(double, double, double, double) ;
  const std::string & reproduction() ;
  polygon_element &
  rhombus(double, double, double, double, double, double, double, double,
          bool = true) ;
  void size(unsigned int, unsigned int) ;
  text_element &
  text(double, double, const std::string &, const text_style &, align_style,
       rotate_style) ;
  polygon_element &
  triangle(double, double, double, double, double, double, bool = true) ;
  void write(std::ostream &) ;
  void write(const std::string &) ;
  unsigned int x_size() ;
  void x_size(unsigned int) ;
  std::pair< double, double > xy_sizes() ;
  unsigned int y_size() ;
  void y_size(unsigned int) ;
  std::string attribution_;  // License requirements for attribution: "permits", "requires", or ↵
"prohibits".
  std::string author_;  // Author(s) name. (Probably == copyright holder).
  std::vector< clip_path_element > clip_paths;  // Points on clip path (used for plot window).
  std::string commercialuse_;  // License requirements for commerical use: "permits", "requires", ↵
or "prohibits".
  int coord_precision_;  // Number of decimal digits precision for output of X and Y coordinates ↵
to SVG XML.
  std::string css_;  // Cascading Style Sheet.
  std::string date_copyright_;  // SVG info on date of copyright claimed.
  std::string derivative_works_;  // License requirements for derivative: "permits", "requires", ↵
or "prohibits".
  std::string distribution_;  // License requirements for distribution: "permits", "requires", ↵
or "prohibits".
  g_element document;  // To hold all group elements of the svg document.
  std::string filename_;  // file SVG XML written to.
  std::string holder_copyright_;  // SVG info on holder of copyright (probably == author, but ↵
could be an institution).
  std::string image_desc_;  // SVG image description (appears in the SVG file header as.
  bool is_boost_license_;  // If true, to include the Boost license as a XML comment.
  bool is_license_;  // If true, to include Creative Commons license as metadata:.
  std::string reproduction_;  // License requirements for reproduction: "permits", "requires", ↵
or "prohibits".
  std::string title_document_;  // SVG document title (appears in the SVG file header as.
  unsigned int x_size_;  // SVG image X-axis size (in SVG units (default pixels).
  unsigned int y_size_;  // SVG image Y-axis size (in SVG units (default pixels).
};
```

Description

Class to add basic Scalable Vector Graph XML graph elements: point, path, line, circle, rect, polygon and text to SVG images, including metadata like author, copyright and license, and to output the final image as SVG XML to stream or file. (Also to handle plot stylesheets - but not implemented yet).

svg public construct/copy/destruct

1.
```
svg(const svg & rhs);
```

Define image size copy constructor.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

2. ```
svg();
```

Define default constructor.

### svg private member functions

1. ```
void write_css(std::ostream & s_out) ;
```

Output CSS (Cascading Style Sheet) - (not yet used or implemented).

2. ```
void write_document(std::ostream & s_out) ;
```

Output all of the image to the svg document (Internal function)

Output all clip paths that define a region of the output device to which paint can be applied.

3. ```
void write_header(std::ostream & s_out) ;
```

Output the DTD SVG 1.1 header into the svg g_element document.

svg public member functions

1. ```
g_element & add_g_element() ;
```

Add information about a group element to the document. Increments the size of the array of g_elements, returned by g_element.size().

return reference to the added group element.

2. ```
const std::string & attribution() ;
```

Returns: license attribution requirement.

3. ```
const std::string & author() ;
```

Returns: author of the document (for header as <author>).

4. ```
void author(const std::string a) ;
```

Set author for the document (default is copyright_holder).

5. ```
bool boost_license_on() ;
```

Return true if a boost license has been requested in the svg header as comment.

6. ```
void boost_license_on(bool l) ;
```

Set (or not) to include Boost license text in svg header as comment.

7. ```
circle_element & circle(double x, double y, unsigned int radius = 5) ;
```

push_back information about a circle to the document. 'circle' element defines a circle centered at (x1, y1) and its radius.



XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/
```

8.
```
clip_path_element &
clip_path(const rect_element & rect, const std::string & id) ;
```

Rectangle outside which 'painting' is 'clipped' so doesn't show.

Returns: Reference to clip_path element.

9.
```
const std::string & commercialuse() ;
```

Returns: license commercial use requirement.

10.
```
int coord_precision() ;
```

Returns: decimal digits to be output for X and Y coordinates.

11.
```
void coord_precision(int digits) ;
```

Set decimal digits to be output for X and Y coordinates.

Default stream precision 6 decimal digits is probably excessive. 4.1 Basic data types, integer or float in decimal or scientific (using e format). 3 or 4 probably enough if image size is under 1000 x 1000. This will reduce .svg file sizes significantly for curves represented with many data points. For example, if a curve is shown using 100 points, reducing to precision(3) from 6 will reduce file size by 300 bytes. So a default of 3 is used in the default constructor above, but can be changed using this function. Used in svg.write below and so applies to all the svg document.

12.
```
const std::string copyright_date() ;
```

Returns: document title for the document (for header as <copyright_date>).

13.
```
void copyright_date(const std::string copyright_date) ;
```

Set document title for the document (for header as <copyright_date>).

14.
```
const std::string copyright_holder() ;
```

\return document title for the document (for header as <copyright_holder>).

15.
```
void copyright_holder(const std::string copyright_holder) ;
```

Set document title for the document (for header as <copyright_holder>).

16.
```
const std::string & description() ;
```

Returns: description of the document (for header as <desc>).

17.
```
void description(const std::string d) ;
```

Write description to the document (for header as <desc> ... </desc>).

18.
```
const std::string & distribution() ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Returns: license distribution requirement.

19.
```
unsigned int document_size() ;
```

How many group elements groups have been added to the document.

20.
```
const std::string document_title() ;
```

Returns: document title for the document (for header as <title>).

21.
```
void document_title(const std::string title) ;
```

Set document title for the document (for header as <title> ... </title>).

22.
```
ellipse_element & ellipse(double rx, double ry, double cx, double cy) ;
```

push_back information about a ellipse to the document. 'ellipse' element defines a ellipse centered at point (x1, y1) and its two radii.

23.
```
g_element & g(int i) ;
```

from array of g_elements, indexed by group type, PLOT_BACKGROUND, PLOT_WINDOW_BACKGROUND, ... SVG_PLOT_DOC_CHILDREN,

return reference to the ith group element.

24.
```
polygon_element &
hexagon(double x1, double y1, double x2, double y2, double x3, double y3,
        double x4, double y4, double x5, double y5, double x6, double y6,
        bool f = true) ;
```

push_back the coordinate of the points of a complete hexagon to the document.

25.
```
const std::string image_filename() ;
```

Returns: image filename for the document (for header as <filename>).

26.
```
void image_filename(const std::string filename) ;
```

Set image filename for the document (for header as <filename>).

27.
```
void license(const std::string reproduction = "permits",
             const std::string distribution = "permits",
             const std::string attribution = "requires",
             const std::string commercialuse = "permits",
             const std::string derivative = "permits") ;
```

Set several license requirements for the svg document.

If any are set, then a license is wanted, so svg::is_license is set true. This can be changed using function license_on().

28.
```
bool license_on() ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Return true if a license has been requested for svg header metatadata.

29.
```
void license_on(bool l) ;
```

Set (or not) license using all requirement (default permits). Implicitly set by setting a license requirement using license above.

30.
```
line_element & line(double x1, double y1, double x2, double y2) ;
```

Add (push_back) information about a line to the document. 'line' element defines a line segment that starts at one point (x1, y1) and ends at another (x2, y2).

31.
```
path_element & path() ;
```

Construct an empty path, ready for additions with chainable functions M., L., ...

Returns: reference to path element.

32.
```
polygon_element &
pentagon(double x1, double y1, double x2, double y2, double x3, double y3,
         double x4, double y4, double x5, double y5, bool f = true) ;
```

push_back the five coordinates complete pentagon to the document.

33.
```
polygon_element & polygon(std::vector< poly_path_point > & v, bool f = true) ;
```

push_back a complete many-sided polygon to the document with vertices specified as a vector of path_points.

34.
```
polygon_element & polygon(double x, double y, bool f = true) ;
```

push_back info about 1st point of a polygon shape (add others later with .P(x, y)).

35.
```
polyline_element & polyline(std::vector< poly_path_point > & v) ;
```

push_back a complete many-sided polygon to the document, from a vector of path_points.

36.
```
polyline_element & polyline(double x1, double y1, double x2, double y2) ;
```

push_back info about the 1st & 2nd point of a polyline (add others later with .P(x, y)).

37.
```
polyline_element & polyline(double x, double y) ;
```

push_back info about the 1st point of a polyline (add others later with .P(x, y)).

38.
```
rect_element & rect(double x1, double y1, double x2, double y2) ;
```

push_back information about a rectangle to the document. 'lRect' element defines a rect segment with one point (x1, y1) and opposite vertex is (x2, y2).

39.
```
const std::string & reproduction() ;
```

Returns: license reproduction requirement.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

40.
```
polygon_element &
rhombus(double x1, double y1, double x2, double y2, double x3, double y3,
        double x4, double y4, bool f = true) ;
```

push_back the four coordinate of a complete rhombus to the document.

41.
```
void size(unsigned int x, unsigned int y) ;
```

Set both X and Y image size (SVG units, default pixels).

42.
```
text_element &
text(double x, double y, const std::string & text, const text_style & style,
     align_style align, rotate_style rotate) ;
```

push_back information about text to the document, with location, style, alignment & rotation.

43.
```
polygon_element &
triangle(double x1, double y1, double x2, double y2, double x3, double y3,
         bool f = true) ;
```

push_back a complete triangle to the document.

44.
```
void write(std::ostream & s_out) ;
```

Write whole .svg 'file' contents to stream (perhaps a file).

45.
```
void write(const std::string & filename) ;
```

Write whole .svg 'file' contents to file.

46.
```
unsigned int x_size() ;
```

Returns: X-axis (horizontal width) SVG image size.

47.
```
void x_size(unsigned int x) ;
```

Set X-axis (horizontal) image size.

48.
```
std::pair< double, double > xy_sizes() ;
```

Returns: both X and Y sizes (horizontal width and vertical height) of the SVG image.

49.
```
unsigned int y_size() ;
```

Returns: Y-axis (vertical height) SVG image size.

50.
```
void y_size(unsigned int y) ;
```

Set Y-axis (vertical) image size.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Global package_info

boost::svg::package_info — Default SVG package information about this program that produced the SVG image (not the image itself).

Synopsis

```
// In header: <boost/svg_plot/svg.hpp>

static const std::string package_info;
```

Description

Inserted as a SVG comment, for example "<!-- Demo of 1D plot features. --> \n" and also as a <desc> http://www.w3.org/TR/SVG/struct.html#DescriptionAndTitleElements 5.4 The 'desc' and 'title' elements

Header <boost/svg_plot/svg_1d_plot.hpp>

Create 1D plots in Scalable Vector Graphic (SVG) format.

Provides svg_1d_plot data and function to create plots, and svg_1d_plot_series to allow data values to be added. Very many functions allow fine control of the appearance and layout of plots and data markers. Items common to 1D and 2D use axis_plot_frame.

Mar 2009

Jacob Voytko & Paul A. Bristow

```
namespace boost {
  namespace svg {
    class svg_1d_plot;
    class svg_1d_plot_series;
    const std::string strip_e0s(std::string);
  }
}
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Class svg_1d_plot

boost::svg::svg_1d_plot — Hold all data about a plot, and functions to get and set.

Synopsis

```
// In header: <boost/svg_plot/svg_1d_plot.hpp>


class svg_1d_plot {
public:
  // construct/copy/destruct
  svg_1d_plot();

  // public member functions
  svg_1d_plot & autoscale(bool) ;
  bool autoscale() ;
  svg_1d_plot & autoscale_check_limits(bool) ;
  bool autoscale_check_limits() ;
  svg_1d_plot & autoscale_plusminus(double) ;
  double autoscale_plusminus() ;
  bool axes_on() ;
  svg_1d_plot & axes_on(bool) ;
  svg_color background_border_color() ;
  svg_1d_plot & background_border_color(const svg_color &) ;
  double background_border_width() ;
  svg_1d_plot & background_border_width(double) ;
  svg_1d_plot & background_color(const svg_color &) ;
  svg_color background_color() ;
  svg_1d_plot & boost_license_on(bool) ;
  bool boost_license_on() ;
  void calculate_plot_window() ;
  void calculate_transform() ;
  int coord_precision() ;
  svg_1d_plot & coord_precision(int) ;
  const std::string copyright_date() ;
  svg_1d_plot & copyright_date(const std::string) ;
  const std::string copyright_holder() ;
  svg_1d_plot & copyright_holder(const std::string) ;
  double data_lines_width() ;
  svg_1d_plot & data_lines_width(double) ;
  const std::string & description() ;
  svg_1d_plot & description(const std::string) ;
  std::string document_title() ;
  svg_1d_plot & document_title(const std::string) ;
  void draw_axes() ;
  svg_1d_plot &
  draw_line(double, double, double, double, const svg_color & = black) ;
  svg_1d_plot &
  draw_note(double, double, std::string, rotate_style = horizontal,
            align_style = center_align, const svg_color & = black,
            text_style & = no_style) ;
  svg_1d_plot &
  draw_plot_curve(double, double, double, double, double, double,
                  const svg_color & = black) ;
  svg_1d_plot &
  draw_plot_line(double, double, double, double, const svg_color & = black) ;
  double image_border_margin() ;
  svg_1d_plot & image_border_margin(double) ;
  double image_border_width() ;
  svg_1d_plot & image_border_width(double) ;
  std::pair< double, double > image_size() ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
svg_1d_plot & image_x_size(unsigned int) ;
unsigned int image_x_size() ;
svg_1d_plot & image_y_size(unsigned int) ;
unsigned int image_y_size() ;
svg_color legend_background_color() ;
svg_1d_plot & legend_background_color(const svg_color &) ;
svg_color legend_border_color() ;
svg_1d_plot & legend_border_color(const svg_color &) ;
const std::pair< double, double > legend_bottom_right() ;
bool legend_box_fill_on() ;
svg_color legend_color() ;
svg_1d_plot & legend_color(const svg_color &) ;
const std::string & legend_font_family() ;
svg_1d_plot & legend_font_family(const std::string &) ;
const std::string & legend_font_weight() ;
svg_1d_plot & legend_font_weight(const std::string &) ;
int legend_header_font_size() ;
svg_1d_plot & legend_header_font_size(int) ;
bool legend_lines() ;
svg_1d_plot & legend_lines(bool) ;
bool legend_on() ;
svg_1d_plot & legend_on(bool) ;
bool legend_outside() ;
legend_places legend_place() ;
svg_1d_plot & legend_place(legend_places) ;
const std::string legend_title() ;
svg_1d_plot & legend_title(const std::string) ;
unsigned int legend_title_font_size() ;
svg_1d_plot & legend_title_font_size(unsigned int) ;
const std::pair< double, double > legend_top_left() ;
svg_1d_plot & legend_top_left(double, double) ;
double legend_width() ;
svg_1d_plot & legend_width(double) ;
svg_1d_plot &
license(std::string = "permits", std::string = "permits",
        std::string = "requires", std::string = "permits",
        std::string = "permits") ;
const std::string license_attribution() ;
const std::string license_commercialuse() ;
const std::string license_distribution() ;
svg_1d_plot & license_on(bool) ;
bool license_on() ;
const std::string license_reproduction() ;
svg_color limit_color() ;
svg_1d_plot & limit_color(const svg_color &) ;
svg_color limit_fill_color() ;
svg_1d_plot & limit_fill_color(const svg_color &) ;
svg_color one_sd_color() ;
svg_1d_plot & one_sd_color(const svg_color &) ;
template<class T , class U >
  svg_1d_plot_series &
  plot(const T &, const std::string & = "", U = double_1d_convert) ;
template<class T , class U >
  svg_1d_plot_series &
  plot(const T &, const T &, const std::string & = "",
       U = double_1d_convert) ;
template<class T >
  svg_1d_plot_series & plot(const T &, const T &, const std::string & = "") ;
template<class T >
  svg_1d_plot_series & plot(const T &, const std::string & = "") ;
svg_color plot_background_color() ;
svg_1d_plot & plot_background_color(const svg_color &) ;
svg_color plot_border_color() ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
svg_1d_plot & plot_border_color(const svg_color &) ;
svg_1d_plot & plot_border_width(double) ;
double plot_border_width() ;
bool plot_window_on() ;
svg_1d_plot & plot_window_on(bool) ;
std::pair< double, double > plot_window_x() ;
svg_1d_plot & plot_window_x(double, double) ;
double plot_window_x_left() ;
double plot_window_x_right() ;
std::pair< double, double > plot_window_y() ;
svg_1d_plot & plot_window_y(double, double) ;
double plot_window_y_bottom() ;
double plot_window_y_top() ;
void set_ids() ;
svg_1d_plot & size(unsigned int, unsigned int) ;
svg_color three_sd_color() ;
svg_1d_plot & three_sd_color(const svg_color &) ;
const std::string title() ;
svg_1d_plot & title(const std::string) ;
svg_color title_color() ;
svg_1d_plot & title_color(const svg_color &) ;
align_style title_font_alignment() ;
svg_1d_plot & title_font_alignment(align_style) ;
const std::string & title_font_decoration() ;
svg_1d_plot & title_font_decoration(const std::string &) ;
const std::string & title_font_family() ;
svg_1d_plot & title_font_family(const std::string &) ;
int title_font_rotation() ;
svg_1d_plot & title_font_rotation(rotate_style) ;
unsigned int title_font_size() ;
svg_1d_plot & title_font_size(unsigned int) ;
const std::string & title_font_stretch() ;
svg_1d_plot & title_font_stretch(const std::string &) ;
const std::string & title_font_style() ;
svg_1d_plot & title_font_style(const std::string &) ;
const std::string & title_font_weight() ;
svg_1d_plot & title_font_weight(const std::string &) ;
bool title_on() ;
svg_1d_plot & title_on(bool) ;
svg_color two_sd_color() ;
svg_1d_plot & two_sd_color(const svg_color &) ;
void update_image() ;
svg_1d_plot & write(std::ostream &) ;
svg_1d_plot & write(const std::string &) ;
double x_auto_max_value() ;
double x_auto_min_value() ;
double x_auto_tick_interval() ;
int x_auto_ticks() ;
svg_1d_plot & x_autoscale(const T &, const T &) ;
svg_1d_plot & x_autoscale(const T &) ;
svg_1d_plot & x_autoscale(std::pair< double, double >) ;
svg_1d_plot & x_autoscale(bool) ;
bool x_autoscale() ;
svg_color x_axis_color() ;
svg_1d_plot & x_axis_color(const svg_color &) ;
svg_color x_axis_label_color() ;
svg_1d_plot & x_axis_label_color(const svg_color &) ;
bool x_axis_on() ;
svg_1d_plot & x_axis_on(bool) ;
const std::string x_axis_position() ;
double x_axis_width() ;
svg_1d_plot & x_axis_width(double) ;
svg_1d_plot &
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
x_decor(const std::string &, const std::string & = "",
        const std::string & = "") ;
svg_color x_df_color() ;
svg_1d_plot & x_df_color(const svg_color &) ;
bool x_df_on() ;
svg_1d_plot & x_df_on(bool) ;
std::string x_label() ;
svg_1d_plot & x_label(const std::string &) ;
svg_color x_label_color() ;
svg_1d_plot & x_label_color(const svg_color &) ;
const std::string & x_label_font_family() ;
svg_1d_plot & x_label_font_family(const std::string &) ;
unsigned int x_label_font_size() ;
svg_1d_plot & x_label_font_size(unsigned int) ;
bool x_label_on() ;
svg_1d_plot & x_label_on(bool) ;
std::string x_label_units() ;
svg_1d_plot & x_label_units(const std::string &) ;
bool x_label_units_on() ;
svg_1d_plot & x_label_units_on(bool) ;
double x_label_width() ;
svg_1d_plot & x_label_width(double) ;
svg_1d_plot & x_labels_strip_e0s(bool) ;
svg_color x_major_grid_color() ;
svg_1d_plot & x_major_grid_color(const svg_color &) ;
bool x_major_grid_on() ;
svg_1d_plot & x_major_grid_on(bool) ;
double x_major_grid_width() ;
svg_1d_plot & x_major_grid_width(double) ;
double x_major_interval() ;
svg_1d_plot & x_major_interval(double) ;
rotate_style x_major_label_rotation() ;
svg_1d_plot & x_major_label_rotation(rotate_style) ;
int x_major_labels_side() ;
svg_1d_plot & x_major_labels_side(int) ;
double x_major_tick() ;
svg_1d_plot & x_major_tick(double) ;
svg_color x_major_tick_color() ;
svg_1d_plot & x_major_tick_color(const svg_color &) ;
double x_major_tick_length() ;
svg_1d_plot & x_major_tick_length(double) ;
double x_major_tick_width() ;
svg_1d_plot & x_major_tick_width(double) ;
double x_max() ;
svg_1d_plot & x_max(double) ;
double x_min() ;
svg_1d_plot & x_min(double) ;
int x_min_ticks() ;
svg_1d_plot & x_min_ticks(int) ;
svg_color x_minor_grid_color() ;
svg_1d_plot & x_minor_grid_color(const svg_color &) ;
bool x_minor_grid_on() ;
svg_1d_plot & x_minor_grid_on(bool) ;
double x_minor_grid_width() ;
svg_1d_plot & x_minor_grid_width(double) ;
svg_1d_plot & x_minor_interval(double) ;
double x_minor_interval() ;
svg_color x_minor_tick_color() ;
svg_1d_plot & x_minor_tick_color(const svg_color &) ;
double x_minor_tick_length() ;
svg_1d_plot & x_minor_tick_length(double) ;
double x_minor_tick_width() ;
svg_1d_plot & x_minor_tick_width(double) ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
unsigned int x_num_minor_ticks() ;
svg_1d_plot & x_num_minor_ticks(unsigned int) ;
svg_color x_plusminus_color() ;
svg_1d_plot & x_plusminus_color(const svg_color &) ;
bool x_plusminus_on() ;
svg_1d_plot & x_plusminus_on(bool) ;
const std::string x_prefix() ;
std::pair< double, double > x_range() ;
svg_1d_plot & x_range(double, double) ;
const std::string x_separator() ;
int x_steps() ;
svg_1d_plot & x_steps(int) ;
const std::string x_suffix() ;
bool x_ticks_down_on() ;
svg_1d_plot & x_ticks_down_on(bool) ;
int x_ticks_on_window_or_axis() ;
svg_1d_plot & x_ticks_on_window_or_axis(int) ;
bool x_ticks_up_on() ;
svg_1d_plot & x_ticks_up_on(bool) ;
svg_color x_ticks_values_color() ;
svg_1d_plot & x_ticks_values_color(const svg_color &) ;
const std::string & x_ticks_values_font_family() ;
svg_1d_plot & x_ticks_values_font_family(const std::string &) ;
unsigned int x_ticks_values_font_size() ;
svg_1d_plot & x_ticks_values_font_size(unsigned int) ;
std::ios_base::fmtflags x_ticks_values_ioflags() ;
svg_1d_plot & x_ticks_values_ioflags(std::ios_base::fmtflags) ;
int x_ticks_values_precision() ;
svg_1d_plot & x_ticks_values_precision(int) ;
double x_tight() ;
svg_1d_plot & x_tight(double) ;
unsigned int x_value_font_size() ;
svg_1d_plot & x_value_font_size(unsigned int) ;
int x_value_ioflags() ;
svg_1d_plot & x_value_ioflags(int) ;
int x_value_precision() ;
svg_1d_plot & x_value_precision(int) ;
svg_color x_values_color() ;
svg_1d_plot & x_values_color(const svg_color &) ;
const std::string & x_values_font_family() ;
svg_1d_plot & x_values_font_family(const std::string &) ;
unsigned int x_values_font_size() ;
svg_1d_plot & x_values_font_size(unsigned int) ;
std::ios_base::fmtflags x_values_ioflags() ;
svg_1d_plot & x_values_ioflags(std::ios_base::fmtflags) ;
bool x_values_on() ;
svg_1d_plot & x_values_on(bool) ;
int x_values_precision() ;
svg_1d_plot & x_values_precision(int) ;
int x_values_rotation() ;
svg_1d_plot & x_values_rotation(rotate_style) ;
bool x_with_zero() ;
svg_1d_plot & x_with_zero(bool) ;
svg_color y_axis_color() ;
svg_1d_plot & y_axis_color(const svg_color &) ;
bool y_axis_on() ;
svg_1d_plot & y_axis_on(bool) ;
std::string y_label() ;
svg_1d_plot & y_label(const std::string &) ;
svg_color y_label_color() ;
svg_1d_plot & y_label_color(const svg_color &) ;
std::string y_label_units() ;
svg_1d_plot & y_label_units(const std::string &) ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
  bool y_labels_strip_e0s() ;
  double y_minor_interval() ;

  // protected member functions
  void adjust_limits(double &, double &) ;
  void clear_all() ;
  void clear_background() ;
  void clear_grids() ;
  void clear_legend() ;
  void clear_plot_background() ;
  void clear_points() ;
  void clear_title() ;
  void clear_x_axis() ;
  void clear_y_axis() ;
  void draw_legend() ;
  void draw_plot_point(double, double, g_element &, plot_point_style &, unc,
                       unc) ;
  void draw_plot_point_value(double, double, g_element &, value_style &,
                             plot_point_style &, unc) ;
  void draw_plot_point_values(double, double, g_element &, g_element &,
                              const value_style &, const value_style &, unc,
                              unc) ;
  void draw_title() ;
  void draw_x_axis() ;
  void draw_x_axis_label() ;
  void draw_x_major_tick(double, path_element &, path_element &) ;
  void draw_x_minor_tick(double, path_element &, path_element &) ;
  void place_legend_box() ;
  void size_legend_box() ;
  std::string sv(double, const value_style &, bool = false) ;
  void transform_point(double &, double &) ;
  void transform_x(double &) ;
  void transform_y(double &) ;
  text_style a_style_;  // Default text style that contains font size & type etc.
  bool autoscale_check_limits_;  // If true, then check autoscale values for infinity, NaN, max, ↵
min.
  double autoscale_plusminus_;  // For uncertain values, allow for plusminus ellipses showing ↵
67%, 95% and 99% confidence limits.
 < For example, if a max value is 1.2 +or- 0.02, then 1.4 will be used for autoscaling the max↵
imum.
 < Similarly, if a min value is 1.2 +or- 0.02, then 1.0 will be used for autoscaling the minimum.
  svg image_;  // Stored so as to avoid rewriting style information constantly.
  box_style image_border_;  // rectangular border of all image width, color...
  double legend_bottom_;  // bottom of legend box.
  box_style legend_box_;  // rectangular box of legend width, color...
  text_element legend_header_;  // legend box header or title (if any).
  double legend_height_;  // Height of legend box (in pixels). < Size of legend box is controlled ↵
by its contents, < but helpful to store computed coordinates.
  double legend_left_;  // Left of legend box.
  bool legend_lines_;  // If true, include data colored line type in legend box.
  size_t legend_longest_;  // longest (both header & data) string in legend box,
  bool legend_on_;  // If true include a legend box.
  legend_places legend_place_;  // Place for any legend box.
  double legend_right_;  // SVG Coordinates of right of legend box,.
  text_style legend_style_;  // style (font etc of legend.
  double legend_top_;  // Top of legend box.
  double legend_width_;  // Width of legend box (pixels).
  bool outside_legend_on_;  // If true, place legend box outside the plot window.
  double plot_bottom_;  // svg bottom of plot window (calculate_plot_window() sets these values).
  double plot_left_;  // svg left of plot window (calculate_plot_window() sets these values).
  double plot_right_;  // svg right of plot window (calculate_plot_window() sets these values).
  double plot_top_;  // svg top of plot window (calculate_plot_window() sets these values).
  box_style plot_window_border_;  // rectangular border of plot window width, color...
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
  std::string plot_window_clip_;  // = "clip_plot_window" id for clippath < ht↵
tp://www.w3.org/TR/SVG/masking.html#ClipPathElement 14.1 Introduction < clipping paths, which ↵
uses any combination of 'path', 'text' and basic shapes < to serve as the outline where everything ↵
on the "inside" of the outline < is allowed to show through but everything on the outside is ↵
masked out. < So the plot_window_clip_ limits display to a plot_window rectangle.
  bool plot_window_on_;  // Use a separate plot window (not whole image).
  text_style point_symbols_style_;  // Used for data point marking.
  std::vector< svg_1d_plot_series > serieses_;  // The (perhaps several) series of data points ↵
for transformation. < These are sorted into two vectors for normal and abnormal (max, inf and ↵
NaN).
  double text_margin_;  // Marginal space around text items like title,.
  text_element title_info_;  // Title of whole plot.
  bool title_on_;  // If true include a title for the whole plot.
  text_style title_style_;  // style (font etc) of title.
  text_style value_style_;  // Used for data point value label.
  double x_auto_max_value_;  // X maximum value calculated by autoscaling.
  double x_auto_min_value_;  // X minimum value calculated by autoscaling.
  double x_auto_tick_interval_;  // X axis tick major interval.
  int x_auto_ticks_;  // Number of X axis ticks.
  bool x_autoscale_;  // If true, use any autoscale values for scaling the X axis.
  axis_line_style x_axis_;  // style of X axis line.
  text_style x_axis_label_style_;  // style of X axis label.
  int x_axis_position_;
  bool x_include_zero_;  // If autoscaled, include zero.
  text_element x_label_info_;  // X-axis label, Example: "length of widget".
  int x_min_ticks_;  // If autoscaled, set a minimum number of ticks.
  double x_scale_;  // scale used for transform from Cartesian to SVG coordinates.
  double x_shift_;  // shift from SVG origin is top left, Cartesian is bottom right.
  int x_steps_;  // If autoscaled, set any prescaling to decimal 1, 2, 5, 10 etc.
  ticks_labels_style x_ticks_;  // style of X axis tick value labels.
  bool x_ticks_on_;
  double x_tight_;  // How much a value can go beyond the tick value before another tick is re↵
quired.
  text_element x_units_info_;  // For example, to display, "length (meter)".
  text_element x_value_label_info_;  // X-axis tick value label, for example: "1.2" or "1.2e1".
  text_style x_value_label_style_;  // style of X ticks value label.
  bool x_values_on_;  // values of data are shown by markers.
  value_style x_values_style_;  // Used for data point value marking.
  bool y_autoscale_;  // Always false for 1-D plot because Y axis is not autoscaled.
  axis_line_style y_axis_;  // style of Y axis line.
  text_style y_axis_label_style_;  // Not used for 1D but needed by axis_plot_frame.hpp.
  double y_scale_;  // scale used for transform from Cartesian to SVG coordinates.
  double y_shift_;  // shift from SVG origin is top left, Cartesian is bottom right.
  ticks_labels_style y_ticks_;  // style of Y axis tick value labels. (Meaningless for 1D but ↵
added to permit shared code!)
  text_style y_value_label_style_;  // Not used for 1D but needed by axis_plot_frame.hpp.
};
```

Description

All data about a plot, and functions to get and set appearance. axis_plot_frame.hpp contains functions common to 1 and 2-D. See also svg_2d_plot.hpp for 2-D version.

`svg_1d_plot` public construct/copy/destruct

1.
```
svg_1d_plot();
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

`svg_1d_plot` public member functions

1.
```
svg_1d_plot & autoscale(bool b) ;
```

2.
```
bool autoscale() ;
```

3.
```
svg_1d_plot & autoscale_check_limits(bool b) ;
```

4.
```
bool autoscale_check_limits() ;
```

5.
```
svg_1d_plot & autoscale_plusminus(double) ;
```

6.
```
double autoscale_plusminus() ;
```

7.
```
bool axes_on() ;
```

8.
```
svg_1d_plot & axes_on(bool is) ;
```

9.
```
svg_color background_border_color() ;
```

10.
```
svg_1d_plot & background_border_color(const svg_color & col) ;
```

11.
```
double background_border_width() ;
```

12.
```
svg_1d_plot & background_border_width(double w) ;
```

13.
```
svg_1d_plot & background_color(const svg_color & col) ;
```

14.
```
svg_color background_color() ;
```

15.
```
svg_1d_plot & boost_license_on(bool l) ;
```

16.
```
bool boost_license_on() ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

17.
```
void calculate_plot_window() ;
```

Calculate the size and position of the plot window, taking account of the length and font size of axes labels, axis ticks, title and legend box. This version is only for 1-D. All calculations use SVG units, pixels by default.

18.
```
void calculate_transform() ;
```

Calculate scale and shift factors for transform from Cartesian to plot. SVG image is (0, 0) at top left, Cartesian (0, 0) at bottom left.

19.
```
int coord_precision() ;
```

20.
```
svg_1d_plot & coord_precision(int digits) ;
```

21.
```
const std::string copyright_date() ;
```

22.
```
svg_1d_plot & copyright_date(const std::string d) ;
```

23.
```
const std::string copyright_holder() ;
```

24.
```
svg_1d_plot & copyright_holder(const std::string d) ;
```

25.
```
double data_lines_width() ;
```

26.
```
svg_1d_plot & data_lines_width(double width) ;
```

27.
```
const std::string & description() ;
```

28.
```
svg_1d_plot & description(const std::string d) ;
```

29.
```
std::string document_title() ;
```

30.
```
svg_1d_plot & document_title(const std::string d) ;
```

31.
```
void draw_axes() ;
```

Add information to the plot image for X axis lines. (For 1-D, there is, of course, only the horizontal X-axis!)

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

32.
```
svg_1d_plot &
draw_line(double x1, double y1, double x2, double y2,
          const svg_color & col = black) ;
```

33.
```
svg_1d_plot &
draw_note(double x, double y, std::string note, rotate_style rot = horizontal,
          align_style al = center_align, const svg_color & = black,
          text_style & tsty = no_style) ;
```

34.
```
svg_1d_plot &
draw_plot_curve(double x1, double y1, double x2, double y2, double x3,
                double y3, const svg_color & col = black) ;
```

35.
```
svg_1d_plot &
draw_plot_line(double x1, double y1, double x2, double y2,
               const svg_color & col = black) ;
```

36.
```
double image_border_margin() ;
```

37.
```
svg_1d_plot & image_border_margin(double w) ;
```

38.
```
double image_border_width() ;
```

39.
```
svg_1d_plot & image_border_width(double w) ;
```

40.
```
std::pair< double, double > image_size() ;
```

41.
```
svg_1d_plot & image_x_size(unsigned int i) ;
```

42.
```
unsigned int image_x_size() ;
```

43.
```
svg_1d_plot & image_y_size(unsigned int i) ;
```

44.
```
unsigned int image_y_size() ;
```

45.
```
svg_color legend_background_color() ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

46.
```
svg_1d_plot & legend_background_color(const svg_color & col) ;
```

47.
```
svg_color legend_border_color() ;
```

48.
```
svg_1d_plot & legend_border_color(const svg_color & col) ;
```

49.
```
const std::pair< double, double > legend_bottom_right() ;
```

50.
```
bool legend_box_fill_on() ;
```

51.
```
svg_color legend_color() ;
```

52.
```
svg_1d_plot & legend_color(const svg_color & col) ;
```

53.
```
const std::string & legend_font_family() ;
```

54.
```
svg_1d_plot & legend_font_family(const std::string & family) ;
```

55.
```
const std::string & legend_font_weight() ;
```

56.
```
svg_1d_plot & legend_font_weight(const std::string & weight) ;
```

57.
```
int legend_header_font_size() ;
```

58.
```
svg_1d_plot & legend_header_font_size(int size) ;
```

59.
```
bool legend_lines() ;
```

60.
```
svg_1d_plot & legend_lines(bool is) ;
```

61.
```
bool legend_on() ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

62.
```
svg_1d_plot & legend_on(bool cmd) ;
```

63.
```
bool legend_outside() ;
```

64.
```
legend_places legend_place() ;
```

65.
```
svg_1d_plot & legend_place(legend_places l) ;
```

66.
```
const std::string legend_title() ;
```

67.
```
svg_1d_plot & legend_title(const std::string title) ;
```

68.
```
unsigned int legend_title_font_size() ;
```

69.
```
svg_1d_plot & legend_title_font_size(unsigned int size) ;
```

70.
```
const std::pair< double, double > legend_top_left() ;
```

71.
```
svg_1d_plot & legend_top_left(double x, double y) ;
```

72.
```
double legend_width() ;
```

73.
```
svg_1d_plot & legend_width(double width) ;
```

74.
```
svg_1d_plot &
license(std::string repro = "permits", std::string distrib = "permits",
        std::string attrib = "requires", std::string commercial = "permits",
        std::string derivative = "permits") ;
```

75.
```
const std::string license_attribution() ;
```

76.
```
const std::string license_commercialuse() ;
```

77.
```
const std::string license_distribution() ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

78.
```
svg_1d_plot & license_on(bool l) ;
```

79.
```
bool license_on() ;
```

80.
```
const std::string license_reproduction() ;
```

81.
```
svg_color limit_color() ;
```

82.
```
svg_1d_plot & limit_color(const svg_color &) ;
```

83.
```
svg_color limit_fill_color() ;
```

84.
```
svg_1d_plot & limit_fill_color(const svg_color &) ;
```

85.
```
svg_color one_sd_color() ;
```

86.
```
svg_1d_plot & one_sd_color(const svg_color &) ;
```

87.
```
template<class T , class U >
  svg_1d_plot_series &
  plot(const T & container, const std::string & title = "",
       U functor = double_1d_convert) ;
```

Add a data series to the plot, with optional title.

This version of plot includes a functor, allowing other than just convert data values to double(the default).
Returns: a reference to data series just added.

88.
```
template<class T , class U >
  svg_1d_plot_series &
  plot(const T & begin, const T & end, const std::string & title = "",
       U functor = double_1d_convert) ;
```

Add a data series to the plot, with optional title. (Version with custom functor, rather than to double). Note that this version permits a partial range, begin to end, of the container to be used.
Returns: a reference to data series just added.

89.
```
template<class T >
  svg_1d_plot_series &
  plot(const T & begin, const T & end, const std::string & title = "") ;
```

Add a data series to the plot (by default, converting to unc doubles), with optional title.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Note that this version permits a partial range, begin to end, of the container to be used. For example: my_1d_plot.plot(my_data.begin(), my_data.end(), "My container"); my_1d_plot.plot(&my_data[1], &my_data[4], "my_data 1 to 4"); // Add part of data series. Care: last == end which is one past the last, so this only does 1, 2 & 3 - *not* 4!
Returns: a reference to data series just added.

90.
```
template<class T >
  svg_1d_plot_series &
  plot(const T & container, const std::string & title = "") ;
```

Add a data series to the plot (by default, converting to unc doubles), with optional title. Note that this version assumes that *ALL* the data values in the container are used.

Returns: Reference to data series just added.

91.
```
svg_color plot_background_color() ;
```

92.
```
svg_1d_plot & plot_background_color(const svg_color & col) ;
```

93.
```
svg_color plot_border_color() ;
```

94.
```
svg_1d_plot & plot_border_color(const svg_color & col) ;
```

95.
```
svg_1d_plot & plot_border_width(double w) ;
```

96.
```
double plot_border_width() ;
```

97.
```
bool plot_window_on() ;
```

98.
```
svg_1d_plot & plot_window_on(bool cmd) ;
```

99.
```
std::pair< double, double > plot_window_x() ;
```

100.
```
svg_1d_plot & plot_window_x(double min_x, double max_x) ;
```

101.
```
double plot_window_x_left() ;
```

102.
```
double plot_window_x_right() ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
std::pair< double, double > plot_window_y() ;
```

```
svg_1d_plot & plot_window_y(double min_y, double max_y) ;
```

```
double plot_window_y_bottom() ;
```

```
double plot_window_y_top() ;
```

```
void set_ids() ;
```

Document ids for use in identifying group elements, for example: <g id = "PLOT_TITLE".../g>

```
svg_1d_plot & size(unsigned int x, unsigned int y) ;
```

```
svg_color three_sd_color() ;
```

```
svg_1d_plot & three_sd_color(const svg_color &) ;
```

```
const std::string title() ;
```

```
svg_1d_plot & title(const std::string title) ;
```

```
svg_color title_color() ;
```

```
svg_1d_plot & title_color(const svg_color & col) ;
```

```
align_style title_font_alignment() ;
```

```
svg_1d_plot & title_font_alignment(align_style alignment) ;
```

```
const std::string & title_font_decoration() ;
```

```
svg_1d_plot & title_font_decoration(const std::string & decoration) ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
const std::string & title_font_family() ;
```

```
svg_1d_plot & title_font_family(const std::string & family) ;
```

```
int title_font_rotation() ;
```

```
svg_1d_plot & title_font_rotation(rotate_style rotate) ;
```

```
unsigned int title_font_size() ;
```

```
svg_1d_plot & title_font_size(unsigned int i) ;
```

```
const std::string & title_font_stretch() ;
```

```
svg_1d_plot & title_font_stretch(const std::string & stretch) ;
```

```
const std::string & title_font_style() ;
```

```
svg_1d_plot & title_font_style(const std::string & style) ;
```

```
const std::string & title_font_weight() ;
```

```
svg_1d_plot & title_font_weight(const std::string & weight) ;
```

```
bool title_on() ;
```

```
svg_1d_plot & title_on(bool cmd) ;
```

```
svg_color two_sd_color() ;
```

```
svg_1d_plot & two_sd_color(const svg_color &) ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
void update_image() ;
```

Calls functions to add all plot information to the image, including

```
svg_1d_plot & write(std::ostream & s_out) ;
```

Write SVG image to the specified std::ostream. This function also is used by the write to file function.

Default stream precision 6 decimal digits is probably excessive. 4.1 Basic data types, integer or float in decimal or scientific (using e format).

- probably enough if image size is under 1000 x 1000. Reduces .svg file sizes significantly for curves represented with many data points. For example, if a curve is shown using 100 points, reducing to coord_precision(3) from default of 6 will reduce file size by 300 bytes.

```
svg_1d_plot & write(const std::string & file) ;
```

Write SVG image to the specified file.

Write SVG image to the specified file, providing the suffix .svg if none given. This function uses the write to ostream function.

```
double x_auto_max_value() ;
```

```
double x_auto_min_value() ;
```

```
double x_auto_tick_interval() ;
```

```
int x_auto_ticks() ;
```

```
svg_1d_plot & x_autoscale(const T & begin, const T & end) ;
```

autoscale X axis using the whole data series.

```
svg_1d_plot & x_autoscale(const T & container) ;
```

autoscale X axis using a pair of doubles.

```
svg_1d_plot & x_autoscale(std::pair< double, double > p) ;
```

```
svg_1d_plot & x_autoscale(bool b) ;
```

```
bool x_autoscale() ;
```

```
svg_color x_axis_color() ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
svg_1d_plot & x_axis_color(const svg_color & col) ;
```

```
svg_color x_axis_label_color() ;
```

```
svg_1d_plot & x_axis_label_color(const svg_color & col) ;
```

```
bool x_axis_on() ;
```

```
svg_1d_plot & x_axis_on(bool is) ;
```

```
const std::string x_axis_position() ;
```

```
double x_axis_width() ;
```

```
svg_1d_plot & x_axis_width(double width) ;
```

```
svg_1d_plot &
x_decor(const std::string & pre, const std::string & sep = "",
        const std::string & suf = "") ;
```

```
svg_color x_df_color() ;
```

```
svg_1d_plot & x_df_color(const svg_color & col) ;
```

```
bool x_df_on() ;
```

```
svg_1d_plot & x_df_on(bool b) ;
```

```
std::string x_label() ;
```

```
svg_1d_plot & x_label(const std::string & str) ;
```

```
svg_color x_label_color() ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
svg_1d_plot & x_label_color(const svg_color & col) ;
```

```
const std::string & x_label_font_family() ;
```

```
svg_1d_plot & x_label_font_family(const std::string & family) ;
```

```
unsigned int x_label_font_size() ;
```

```
svg_1d_plot & x_label_font_size(unsigned int i) ;
```

```
bool x_label_on() ;
```

```
svg_1d_plot & x_label_on(bool cmd) ;
```

```
std::string x_label_units() ;
```

```
svg_1d_plot & x_label_units(const std::string & str) ;
```

```
bool x_label_units_on() ;
```

```
svg_1d_plot & x_label_units_on(bool cmd) ;
```

```
double x_label_width() ;
```

```
svg_1d_plot & x_label_width(double width) ;
```

```
svg_1d_plot & x_labels_strip_e0s(bool cmd) ;
```

```
svg_color x_major_grid_color() ;
```

```
svg_1d_plot & x_major_grid_color(const svg_color & col) ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
bool x_major_grid_on() ;
```

```
svg_1d_plot & x_major_grid_on(bool is) ;
```

```
double x_major_grid_width() ;
```

```
svg_1d_plot & x_major_grid_width(double w) ;
```

```
double x_major_interval() ;
```

```
svg_1d_plot & x_major_interval(double inter) ;
```

```
rotate_style x_major_label_rotation() ;
```

```
svg_1d_plot & x_major_label_rotation(rotate_style rot) ;
```

```
int x_major_labels_side() ;
```

```
svg_1d_plot & x_major_labels_side(int cmd) ;
```

```
double x_major_tick() ;
```

```
svg_1d_plot & x_major_tick(double d) ;
```

```
svg_color x_major_tick_color() ;
```

```
svg_1d_plot & x_major_tick_color(const svg_color & col) ;
```

```
double x_major_tick_length() ;
```

```
svg_1d_plot & x_major_tick_length(double length) ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
double x_major_tick_width() ;
```

```
svg_1d_plot & x_major_tick_width(double width) ;
```

```
double x_max() ;
```

```
svg_1d_plot & x_max(double x) ;
```

```
double x_min() ;
```

```
svg_1d_plot & x_min(double min_x) ;
```

```
int x_min_ticks() ;
```

```
svg_1d_plot & x_min_ticks(int min_ticks) ;
```

```
svg_color x_minor_grid_color() ;
```

```
svg_1d_plot & x_minor_grid_color(const svg_color & col) ;
```

```
bool x_minor_grid_on() ;
```

```
svg_1d_plot & x_minor_grid_on(bool is) ;
```

```
double x_minor_grid_width() ;
```

```
svg_1d_plot & x_minor_grid_width(double w) ;
```

```
svg_1d_plot & x_minor_interval(double interval) ;
```

```
double x_minor_interval() ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
svg_color x_minor_tick_color() ;
```

```
svg_1d_plot & x_minor_tick_color(const svg_color & col) ;
```

```
double x_minor_tick_length() ;
```

```
svg_1d_plot & x_minor_tick_length(double length) ;
```

```
double x_minor_tick_width() ;
```

```
svg_1d_plot & x_minor_tick_width(double width) ;
```

```
unsigned int x_num_minor_ticks() ;
```

```
svg_1d_plot & x_num_minor_ticks(unsigned int num) ;
```

```
svg_color x_plusminus_color() ;
```

```
svg_1d_plot & x_plusminus_color(const svg_color & col) ;
```

```
bool x_plusminus_on() ;
```

```
svg_1d_plot & x_plusminus_on(bool b) ;
```

```
const std::string x_prefix() ;
```

```
std::pair< double, double > x_range() ;
```

```
svg_1d_plot & x_range(double min_x, double max_x) ;
```

```
const std::string x_separator() ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
int x_steps() ;
```

```
svg_1d_plot & x_steps(int steps) ;
```

```
const std::string x_suffix() ;
```

```
bool x_ticks_down_on() ;
```

```
svg_1d_plot & x_ticks_down_on(bool cmd) ;
```

```
int x_ticks_on_window_or_axis() ;
```

```
svg_1d_plot & x_ticks_on_window_or_axis(int cmd) ;
```

```
bool x_ticks_up_on() ;
```

```
svg_1d_plot & x_ticks_up_on(bool cmd) ;
```

```
svg_color x_ticks_values_color() ;
```

```
svg_1d_plot & x_ticks_values_color(const svg_color & col) ;
```

```
const std::string & x_ticks_values_font_family() ;
```

```
svg_1d_plot & x_ticks_values_font_family(const std::string & family) ;
```

```
unsigned int x_ticks_values_font_size() ;
```

```
svg_1d_plot & x_ticks_values_font_size(unsigned int i) ;
```

```
std::ios_base::fmtflags x_ticks_values_ioflags() ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
svg_1d_plot & x_ticks_values_ioflags(std::ios_base::fmtflags f) ;
```

```
int x_ticks_values_precision() ;
```

```
svg_1d_plot & x_ticks_values_precision(int p) ;
```

```
double x_tight() ;
```

```
svg_1d_plot & x_tight(double tight) ;
```

```
unsigned int x_value_font_size() ;
```

```
svg_1d_plot & x_value_font_size(unsigned int i) ;
```

```
int x_value_ioflags() ;
```

```
svg_1d_plot & x_value_ioflags(int flags) ;
```

```
int x_value_precision() ;
```

```
svg_1d_plot & x_value_precision(int digits) ;
```

```
svg_color x_values_color() ;
```

```
svg_1d_plot & x_values_color(const svg_color & col) ;
```

```
const std::string & x_values_font_family() ;
```

```
svg_1d_plot & x_values_font_family(const std::string & family) ;
```

```
unsigned int x_values_font_size() ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
svg_1d_plot & x_values_font_size(unsigned int i) ;
```

```
std::ios_base::fmtflags x_values_ioflags() ;
```

```
svg_1d_plot & x_values_ioflags(std::ios_base::fmtflags f) ;
```

```
bool x_values_on() ;
```

```
svg_1d_plot & x_values_on(bool b) ;
```

```
int x_values_precision() ;
```

```
svg_1d_plot & x_values_precision(int p) ;
```

```
int x_values_rotation() ;
```

```
svg_1d_plot & x_values_rotation(rotate_style rotate) ;
```

```
bool x_with_zero() ;
```

```
svg_1d_plot & x_with_zero(bool b) ;
```

```
svg_color y_axis_color() ;
```

```
svg_1d_plot & y_axis_color(const svg_color & col) ;
```

```
bool y_axis_on() ;
```

```
svg_1d_plot & y_axis_on(bool is) ;
```

```
std::string y_label() ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

26.
```
svg_1d_plot & y_label(const std::string & str) ;
```

27.
```
svg_color y_label_color() ;
```

28.
```
svg_1d_plot & y_label_color(const svg_color & col) ;
```

29.
```
std::string y_label_units() ;
```

30.
```
svg_1d_plot & y_label_units(const std::string & str) ;
```

31.
```
bool y_labels_strip_e0s() ;
```

32.
```
double y_minor_interval() ;
```

`svg_1d_plot` protected member functions

1.
```
void adjust_limits(double & x, double & y) ;
```

If value of a data point reaches limit of max, min, infinity, use the appropriate plot min or max value instead.

2.
```
void clear_all() ;
```

Clear all layers of the plot.

When writing to multiple documents, the contents of the plot may change significantly between. Rather than figuring out what has and has not changed, just erase the contents of the legend, title... in the document and start over.

3.
```
void clear_background() ;
```

Clear the whole image background layer of the SVG plot.

4.
```
void clear_grids() ;
```

Clear the grids layer of the SVG plot.

5.
```
void clear_legend() ;
```

Clear the legend layer of the SVG plot.

6.
```
void clear_plot_background() ;
```

Clear the plot area background layer of the SVG plot.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

7.
```
void clear_points() ;
```

Clear the data points layer of the SVG plot.

8.
```
void clear_title() ;
```

Clear the plot title layer of the SVG plot.

9.
```
void clear_x_axis() ;
```

Clear the X axis layer of the SVG plot.

10.
```
void clear_y_axis() ;
```

Clear the Y axis layer of the SVG plot.

11.
```
void draw_legend() ;
```

Draw the legend border, text header (if any) and marker lines and/or shapes.

12.
```
void draw_plot_point(double x, double y, g_element & g_ptr,
                     plot_point_style & sty, unc ux, unc uy) ;
```

Draw a plot data point marker shape whose size and stroke and fill colors are specified in plot_point_style.

13.
```
void draw_plot_point_value(double x, double y, g_element & g_ptr,
                           value_style & val_style,
                           plot_point_style & point_style, unc uvalue) ;
```

Write one data point (X or Y) value as a string, for example "1.23e-2", near the data point marker. Unecessary e, +, & leading exponent zeros may optionally be stripped, and the position and rotation controlled. Uncertainty estimate, typically standard deviation (half conventional 95% confidence 'plus or minus') may be optionally be appended. Degrees of freedom estimate (number of replicates) may optionally be appended. For example: "3.45 +-0.1(10)" The precision and format (scientific, fixed), and color and font type and size can be controlled too.

Unicode space plusminus glyph.

14.
```
void draw_plot_point_values(double x, double y, g_element & x_g_ptr,
                            g_element & y_g_ptr, const value_style & x_sty,
                            const value_style & y_sty, unc uncx, unc uncy) ;
```

Write the **pair** of data point's X and Y values as a string. If a separator, then both on the same line, for example "1.23, 3.45", or "[5.6, 7.8] X value_style is used to provide the prefix and separator, and Y value_style to provide the suffix. For example, x_style prefix("[X=", and separator ",\<ampersand\>\#x00A0;Y= ", " and Y value_style = "]" will produce a value label like "[X=-1.23, Y=4.56]" Note the need to use a Unicode space <ampersand>#x00A0; for get space for all browsers. For as long a string as this you may need to make the total image size bigger, and to orient the value labels with care. draw_plot_point_values is only when both x and Y pairs are wanted.

Also strip unnecessary e, + and leading exponent zeros, if required.

Unicode space plusminus glyph.

15.
```
void draw_title() ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Draw title (for the whole plot). Update title_info_ with position. Assumes align = center_align Using center_align will ensure that title will center correctly because the render engine does the centering. (Even if the original string is made much longer because it contains Unicode, Greek, math symbols etc, taking about 8 characters per symbol. For example, the Unicode symbol for square root is "√" but takes only about one character width).

16.
```
void draw_x_axis() ;
```

Draw horizontal X-axis line & plot window line to hold, and ticks and grids.

17.
```
void draw_x_axis_label() ;
```

Draw the X-axis label text (for example, length), and append any optional units (for example, km).

18.
```
void draw_x_major_tick(double value, path_element & tick_path,
                       path_element & grid_path) ;
```

Draw major ticks - and grid too if wanted. If major_value_labels_side then value shown beside the major tick.

19.
```
void draw_x_minor_tick(double value, path_element & tick_path,
                       path_element & grid_path) ;
```

Draw X-axis minor ticks, and optional grid. (Value is NOT (yet) shown beside the minor tick).

20.
```
void place_legend_box() ;
```

Place legend box (if required).

21.
```
void size_legend_box() ;
```

Calculate how big the legend box needs to be.

22.
```
std::string sv(double v, const value_style & sty, bool precise = false) ;
```

Strip from double value any unecessary e, +, & leading exponent zeros, reducing "1.200000" to "1.2" or "3.4e1"...

23.
```
void transform_point(double & x, double & y) ;
```

Scale & shift both X & Y to graph Cartesian coordinates.

24.
```
void transform_x(double & x) ;
```

Scale and shift X value only.

25.
```
void transform_y(double & y) ;
```

Scale and shift Y value only.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Class svg_1d_plot_series

boost::svg::svg_1d_plot_series — Holds a series of data values (points) to be plotted.

Synopsis

```
// In header: <boost/svg_plot/svg_1d_plot.hpp>


class svg_1d_plot_series {
public:
  // construct/copy/destruct
  template<class T > svg_1d_plot_series(T, T, const std::string & = "");

  // public member functions
  bool bezier_on() ;
  svg_1d_plot_series & bezier_on(bool) ;
  svg_color fill_color() ;
  svg_1d_plot_series & fill_color(const svg_color &) ;
  svg_1d_plot_series & limit_point_color(const svg_color &) ;
  svg_1d_plot_series & line_color(const svg_color &) ;
  bool line_on() ;
  svg_1d_plot_series & line_on(bool) ;
  double line_width() ;
  svg_1d_plot_series & line_width(double) ;
  size_t series_count() ;
  size_t series_limits_count() ;
  point_shape shape() ;
  svg_1d_plot_series & shape(point_shape) ;
  int size() ;
  svg_1d_plot_series & size(int) ;
  svg_color stroke_color() ;
  svg_1d_plot_series & stroke_color(const svg_color &) ;
  const std::string symbols() ;
  svg_1d_plot_series & symbols(const std::string) ;
  plot_point_style limit_point_style_;  // Default is cone pointing down.
  plot_line_style line_style_;  // No line style for 1-D, only for 2-D.
  plot_point_style point_style_;  // circle, square...
  std::vector< unc > series_;  // Normal 'OK to plot' data values.
  std::vector< double > series_limits_;  // 'limit' values: too big, too small or NaN.
  std::string title_;  // title of data series (to show on legend).
};
```

Description

Scan each data point sorting them into the appropriate std::vectors, normal or not (NaN or infinite). Member functions allow control of data points markers and lines joining them, and their appearance, shape, color and size. Each data series can have a title that can be shown on a legend box with identifying symbols.

`svg_1d_plot_series` public construct/copy/destruct

1.
```
template<class T >
  svg_1d_plot_series(T begin, T end, const std::string & title = "");
```

   Scan each data point between the iterators that are passed, sorting them into the appropriate std::vectors, normal or not (NaN or infinite).

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

`svg_1d_plot_series` public member functions

1. ```
   bool bezier_on() ;
   ```

   Get true if curve if to join data points.

   Returns: true if to draw bezier curved line joining plot points.

2. ```
   svg_1d_plot_series & bezier_on(bool on_) ;
   ```

   Set true for curve joining points for plot point marker(s) (chainable).

   Set true if to draw bezier curved line joining plot points.

3. ```
   svg_color fill_color() ;
   ```

   Get fill color for plot point marker(s).

4. ```
   svg_1d_plot_series & fill_color(const svg_color & col_) ;
   ```

   Set fill color for plot point marker(s) (chainable).

   Set fill color for plot point marker(s) (chainable).

   Returns: reference to svg_1d_plot_series to make chainable.

5. ```
   svg_1d_plot_series & limit_point_color(const svg_color & col_) ;
   ```

   Set stroke color for 'at limits' point marker.

   Set of stroke color of 'at limits' points.

6. ```
   svg_1d_plot_series & line_color(const svg_color & col_) ;
   ```

   Set color for line joining data points (chainable).

   Set color of any line joining plot points.

7. ```
   bool line_on() ;
   ```

   Get true if line is to join data points.

   Returns: true if to draw a line joining plot points.

8. ```
   svg_1d_plot_series & line_on(bool on_) ;
   ```

   Set true for line joining points for plot point marker(s) (chainable).

   Set true if to draw a line joining plot points.

9. ```
   double line_width() ;
   ```

   Get line width for plot point marker(s).

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/
   ```

Returns: width of any line joining plot points.

10.
```
svg_1d_plot_series & line_width(double wid_) ;
```

Set line width for plot point marker(s) (chainable).

Set width of any line joining plot points.

11.
```
size_t series_count() ;
```

Get number of normal data points in this data series.

Number of normal 'OK to plot' data values in data series.

12.
```
size_t series_limits_count() ;
```

Get number of 'at limits' data points in this data series.

Number of 'at limit' values: too big, too small or NaN data values in data series.

13.
```
point_shape shape() ;
```

Get shape for plot point marker(s).

Returns: shape for plot point marker(s).

14.
```
svg_1d_plot_series & shape(point_shape shape_) ;
```

Set shape for plot point marker(s) (chainable).

Set shape for plot point marker(s) (chainable).

Returns: reference to svg_1d_plot_series to make chainable.

15.
```
int size() ;
```

Get size for plot point marker(s).

Returns: size of plot point marker(s).

16.
```
svg_1d_plot_series & size(int size_) ;
```

Set font size for plot point marker(s) (chainable).

Set size of plot point marker(s).

Returns: reference to svg_1d_plot_series to make chainable.

17.
```
svg_color stroke_color() ;
```

Get stroke color for plot point marker(s).

18.
```
svg_1d_plot_series & stroke_color(const svg_color & col_) ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Set stroke color for plot point marker(s) (chainable).

Set stroke color for plot point marker(s) (chainable).

Returns: reference to svg_1d_plot_series to make chainable.

19.
```
const std::string symbols() ;
```

Get sumbols for plot point marker(s).

Returns: symbol for plot point marker(s).

20.
```
svg_1d_plot_series & symbols(const std::string s) ;
```

Set symbol(s) for plot point marker(s) (chainable).

Set symbol for plot point marker(s).

Returns: reference to svg_1d_plot_series to make chainable.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Function strip_e0s

boost::svg::strip_e0s

Synopsis

```
// In header: <boost/svg_plot/svg_1d_plot.hpp>


const std::string strip_e0s(std::string s);
```

Description

Returns: length of trimmed string (perhaps unchanged).

length of trimmed string (perhaps unchanged).

Header <boost/svg_plot/svg_2d_plot.hpp>

Create 2D plots in Scalable Vector Graphic (SVG) format.

Provides svg_2d_plot data and function to create plots, and svg_2d_plot_series to allow data values to be added. Very many functions allow fine control of the appearance and layout of plots, data markers and lines. Items common to 1D and 2D use axis_plot_frame.

Jacob Voytko & Paul A. Bristow

Mar 2009

```
namespace boost {
  namespace svg {
    class svg_2d_plot;
    class svg_2d_plot_series;
  }
}
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Class svg_2d_plot

boost::svg::svg_2d_plot — Provides svg_2d_plot data and functions to create plots. Very many functions allow very fine control of the appearance and layout of plots, data markers and lines.

Synopsis

```
// In header: <boost/svg_plot/svg_2d_plot.hpp>


class svg_2d_plot {
public:
  // construct/copy/destruct
  svg_2d_plot();

  // public member functions
  svg_2d_plot & autoscale(bool) ;
  bool autoscale() ;
  svg_2d_plot & autoscale_check_limits(bool) ;
  bool autoscale_check_limits() ;
  svg_2d_plot & autoscale_plusminus(double) ;
  double autoscale_plusminus() ;
  bool axes_on() ;
  svg_2d_plot & axes_on(bool) ;
  svg_color background_border_color() ;
  svg_2d_plot & background_border_color(const svg_color &) ;
  double background_border_width() ;
  svg_2d_plot & background_border_width(double) ;
  svg_2d_plot & background_color(const svg_color &) ;
  svg_color background_color() ;
  svg_2d_plot & boost_license_on(bool) ;
  bool boost_license_on() ;
  int coord_precision() ;
  svg_2d_plot & coord_precision(int) ;
  const std::string copyright_date() ;
  svg_2d_plot & copyright_date(const std::string) ;
  const std::string copyright_holder() ;
  svg_2d_plot & copyright_holder(const std::string) ;
  double data_lines_width() ;
  svg_2d_plot & data_lines_width(double) ;
  const std::string & description() ;
  svg_2d_plot & description(const std::string) ;
  std::string document_title() ;
  svg_2d_plot & document_title(const std::string) ;
  svg_2d_plot &
  draw_line(double, double, double, double, const svg_color & = black) ;
  svg_2d_plot &
  draw_note(double, double, std::string, rotate_style = horizontal,
            align_style = center_align, const svg_color & = black,
            text_style & = no_style) ;
  svg_2d_plot &
  draw_plot_curve(double, double, double, double, double, double,
                  const svg_color & = black) ;
  svg_2d_plot &
  draw_plot_line(double, double, double, double, const svg_color & = black) ;
  double image_border_margin() ;
  svg_2d_plot & image_border_margin(double) ;
  double image_border_width() ;
  svg_2d_plot & image_border_width(double) ;
  std::pair< double, double > image_size() ;
  svg_2d_plot & image_x_size(unsigned int) ;
  unsigned int image_x_size() ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
svg_2d_plot & image_y_size(unsigned int) ;
unsigned int image_y_size() ;
svg_color legend_background_color() ;
svg_2d_plot & legend_background_color(const svg_color &) ;
svg_color legend_border_color() ;
svg_2d_plot & legend_border_color(const svg_color &) ;
const std::pair< double, double > legend_bottom_right() ;
bool legend_box_fill_on() ;
svg_color legend_color() ;
svg_2d_plot & legend_color(const svg_color &) ;
const std::string & legend_font_family() ;
svg_2d_plot & legend_font_family(const std::string &) ;
const std::string & legend_font_weight() ;
svg_2d_plot & legend_font_weight(const std::string &) ;
int legend_header_font_size() ;
svg_2d_plot & legend_header_font_size(int) ;
bool legend_lines() ;
svg_2d_plot & legend_lines(bool) ;
bool legend_on() ;
svg_2d_plot & legend_on(bool) ;
bool legend_outside() ;
legend_places legend_place() ;
svg_2d_plot & legend_place(legend_places) ;
const std::string legend_title() ;
svg_2d_plot & legend_title(const std::string) ;
unsigned int legend_title_font_size() ;
svg_2d_plot & legend_title_font_size(unsigned int) ;
const std::pair< double, double > legend_top_left() ;
svg_2d_plot & legend_top_left(double, double) ;
double legend_width() ;
svg_2d_plot & legend_width(double) ;
svg_2d_plot &
license(std::string = "permits", std::string = "permits",
        std::string = "requires", std::string = "permits",
        std::string = "permits") ;
const std::string license_attribution() ;
const std::string license_commercialuse() ;
const std::string license_distribution() ;
svg_2d_plot & license_on(bool) ;
bool license_on() ;
const std::string license_reproduction() ;
svg_color limit_color() ;
svg_2d_plot & limit_color(const svg_color &) ;
svg_color limit_fill_color() ;
svg_2d_plot & limit_fill_color(const svg_color &) ;
svg_color one_sd_color() ;
svg_2d_plot & one_sd_color(const svg_color &) ;
template<class T , class U >
  svg_2d_plot_series &
  plot(const T &, const T &, const std::string & = "",
       U = pair_double_2d_convert) ;
template<class T >
  svg_2d_plot_series & plot(const T &, const T &, const std::string & = "") ;
template<class T , class U >
  svg_2d_plot_series &
  plot(const T &, const std::string & = "", U = pair_double_2d_convert) ;
template<class T >
  svg_2d_plot_series & plot(const T &, const std::string & = "") ;
svg_color plot_background_color() ;
svg_2d_plot & plot_background_color(const svg_color &) ;
svg_color plot_border_color() ;
svg_2d_plot & plot_border_color(const svg_color &) ;
svg_2d_plot & plot_border_width(double) ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
double plot_border_width() ;
bool plot_window_on() ;
svg_2d_plot & plot_window_on(bool) ;
std::pair< double, double > plot_window_x() ;
svg_2d_plot & plot_window_x(double, double) ;
double plot_window_x_left() ;
double plot_window_x_right() ;
std::pair< double, double > plot_window_y() ;
svg_2d_plot & plot_window_y(double, double) ;
double plot_window_y_bottom() ;
double plot_window_y_top() ;
svg_2d_plot & size(unsigned int, unsigned int) ;
svg_color three_sd_color() ;
svg_2d_plot & three_sd_color(const svg_color &) ;
const std::string title() ;
svg_2d_plot & title(const std::string) ;
svg_color title_color() ;
svg_2d_plot & title_color(const svg_color &) ;
align_style title_font_alignment() ;
svg_2d_plot & title_font_alignment(align_style) ;
const std::string & title_font_decoration() ;
svg_2d_plot & title_font_decoration(const std::string &) ;
const std::string & title_font_family() ;
svg_2d_plot & title_font_family(const std::string &) ;
int title_font_rotation() ;
svg_2d_plot & title_font_rotation(rotate_style) ;
unsigned int title_font_size() ;
svg_2d_plot & title_font_size(unsigned int) ;
const std::string & title_font_stretch() ;
svg_2d_plot & title_font_stretch(const std::string &) ;
const std::string & title_font_style() ;
svg_2d_plot & title_font_style(const std::string &) ;
const std::string & title_font_weight() ;
svg_2d_plot & title_font_weight(const std::string &) ;
bool title_on() ;
svg_2d_plot & title_on(bool) ;
svg_color two_sd_color() ;
svg_2d_plot & two_sd_color(const svg_color &) ;
svg_2d_plot & write(std::ostream &) ;
svg_2d_plot & write(const std::string &) ;
double x_auto_max_value() ;
double x_auto_min_value() ;
double x_auto_tick_interval() ;
int x_auto_ticks() ;
svg_2d_plot & x_autoscale(const T &, const T &) ;
svg_2d_plot & x_autoscale(const T &) ;
svg_2d_plot & x_autoscale(std::pair< double, double >) ;
svg_2d_plot & x_autoscale(bool) ;
bool x_autoscale() ;
axis_line_style & x_axis() ;
svg_color x_axis_color() ;
svg_2d_plot & x_axis_color(const svg_color &) ;
svg_color x_axis_label_color() ;
svg_2d_plot & x_axis_label_color(const svg_color &) ;
bool x_axis_on() ;
svg_2d_plot & x_axis_on(bool) ;
const std::string x_axis_position() ;
double x_axis_width() ;
svg_2d_plot & x_axis_width(double) ;
svg_2d_plot &
x_decor(const std::string &, const std::string & = "",
        const std::string & = "") ;
svg_color x_df_color() ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
svg_2d_plot & x_df_color(const svg_color &) ;
bool x_df_on() ;
svg_2d_plot & x_df_on(bool) ;
std::string x_label() ;
svg_2d_plot & x_label(const std::string &) ;
svg_color x_label_color() ;
svg_2d_plot & x_label_color(const svg_color &) ;
const std::string & x_label_font_family() ;
svg_2d_plot & x_label_font_family(const std::string &) ;
unsigned int x_label_font_size() ;
svg_2d_plot & x_label_font_size(unsigned int) ;
bool x_label_on() ;
svg_2d_plot & x_label_on(bool) ;
std::string x_label_units() ;
svg_2d_plot & x_label_units(const std::string &) ;
bool x_label_units_on() ;
svg_2d_plot & x_label_units_on(bool) ;
double x_label_width() ;
svg_2d_plot & x_label_width(double) ;
svg_2d_plot & x_labels_strip_e0s(bool) ;
svg_color x_major_grid_color() ;
svg_2d_plot & x_major_grid_color(const svg_color &) ;
bool x_major_grid_on() ;
svg_2d_plot & x_major_grid_on(bool) ;
double x_major_grid_width() ;
svg_2d_plot & x_major_grid_width(double) ;
double x_major_interval() ;
svg_2d_plot & x_major_interval(double) ;
rotate_style x_major_label_rotation() ;
svg_2d_plot & x_major_label_rotation(rotate_style) ;
int x_major_labels_side() ;
svg_2d_plot & x_major_labels_side(int) ;
double x_major_tick() ;
svg_2d_plot & x_major_tick(double) ;
svg_color x_major_tick_color() ;
svg_2d_plot & x_major_tick_color(const svg_color &) ;
double x_major_tick_length() ;
svg_2d_plot & x_major_tick_length(double) ;
double x_major_tick_width() ;
svg_2d_plot & x_major_tick_width(double) ;
double x_max() ;
svg_2d_plot & x_max(double) ;
double x_min() ;
svg_2d_plot & x_min(double) ;
int x_min_ticks() ;
svg_2d_plot & x_min_ticks(int) ;
svg_color x_minor_grid_color() ;
svg_2d_plot & x_minor_grid_color(const svg_color &) ;
bool x_minor_grid_on() ;
svg_2d_plot & x_minor_grid_on(bool) ;
double x_minor_grid_width() ;
svg_2d_plot & x_minor_grid_width(double) ;
svg_2d_plot & x_minor_interval(double) ;
double x_minor_interval() ;
svg_color x_minor_tick_color() ;
svg_2d_plot & x_minor_tick_color(const svg_color &) ;
double x_minor_tick_length() ;
svg_2d_plot & x_minor_tick_length(double) ;
double x_minor_tick_width() ;
svg_2d_plot & x_minor_tick_width(double) ;
unsigned int x_num_minor_ticks() ;
svg_2d_plot & x_num_minor_ticks(unsigned int) ;
svg_color x_plusminus_color() ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
svg_2d_plot & x_plusminus_color(const svg_color &) ;
bool x_plusminus_on() ;
svg_2d_plot & x_plusminus_on(bool) ;
const std::string x_prefix() ;
std::pair< double, double > x_range() ;
svg_2d_plot & x_range(double, double) ;
const std::string x_separator() ;
int x_steps() ;
svg_2d_plot & x_steps(int) ;
const std::string x_suffix() ;
ticks_labels_style & x_ticks() ;
bool x_ticks_down_on() ;
svg_2d_plot & x_ticks_down_on(bool) ;
int x_ticks_on_window_or_axis() ;
svg_2d_plot & x_ticks_on_window_or_axis(int) ;
bool x_ticks_up_on() ;
svg_2d_plot & x_ticks_up_on(bool) ;
svg_color x_ticks_values_color() ;
svg_2d_plot & x_ticks_values_color(const svg_color &) ;
const std::string & x_ticks_values_font_family() ;
svg_2d_plot & x_ticks_values_font_family(const std::string &) ;
unsigned int x_ticks_values_font_size() ;
svg_2d_plot & x_ticks_values_font_size(unsigned int) ;
std::ios_base::fmtflags x_ticks_values_ioflags() ;
svg_2d_plot & x_ticks_values_ioflags(std::ios_base::fmtflags) ;
int x_ticks_values_precision() ;
svg_2d_plot & x_ticks_values_precision(int) ;
double x_tight() ;
svg_2d_plot & x_tight(double) ;
unsigned int x_value_font_size() ;
svg_2d_plot & x_value_font_size(unsigned int) ;
int x_value_ioflags() ;
svg_2d_plot & x_value_ioflags(int) ;
int x_value_precision() ;
svg_2d_plot & x_value_precision(int) ;
svg_color x_values_color() ;
svg_2d_plot & x_values_color(const svg_color &) ;
const std::string & x_values_font_family() ;
svg_2d_plot & x_values_font_family(const std::string &) ;
unsigned int x_values_font_size() ;
svg_2d_plot & x_values_font_size(unsigned int) ;
std::ios_base::fmtflags x_values_ioflags() ;
svg_2d_plot & x_values_ioflags(std::ios_base::fmtflags) ;
bool x_values_on() ;
svg_2d_plot & x_values_on(bool) ;
int x_values_precision() ;
svg_2d_plot & x_values_precision(int) ;
int x_values_rotation() ;
svg_2d_plot & x_values_rotation(rotate_style) ;
bool x_with_zero() ;
svg_2d_plot & x_with_zero(bool) ;
bool xy_autoscale() ;
template<class T > svg_2d_plot & xy_autoscale(const T &) ;
svg_2d_plot & xy_values_on(bool) ;
bool xy_values_on() ;
template<class T > svg_2d_plot & y_autoscale(const T &) ;
template<class T > svg_2d_plot & y_autoscale(const T &, const T &) ;
svg_2d_plot & y_autoscale(std::pair< double, double >) ;
svg_2d_plot & y_autoscale(double, double) ;
svg_2d_plot & y_autoscale(bool) ;
bool y_autoscale() ;
axis_line_style & y_axis() ;
svg_color y_axis_color() ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
svg_2d_plot & y_axis_color(const svg_color &) ;
svg_color y_axis_label_color() ;
svg_2d_plot & y_axis_label_color(const svg_color &) ;
bool y_axis_on() ;
svg_2d_plot & y_axis_on(bool) ;
const std::string y_axis_position() ;
svg_color y_axis_value_color() ;
svg_2d_plot & y_axis_value_color(const svg_color &) ;
double y_axis_width() ;
svg_2d_plot & y_axis_width(double) ;
svg_2d_plot &
y_decor(const std::string &, const std::string & = "",
        const std::string & = "") ;
const svg_color y_df_color() ;
svg_2d_plot & y_df_color(const svg_color &) ;
svg_2d_plot & y_df_on(bool) ;
bool y_df_on() ;
std::string y_label() ;
svg_2d_plot & y_label(const std::string &) ;
std::string y_label_axis() ;
svg_2d_plot & y_label_axis(const std::string &) ;
svg_color y_label_color() ;
svg_2d_plot & y_label_color(const svg_color &) ;
const std::string & y_label_font_family() ;
svg_2d_plot & y_label_font_family(const std::string &) ;
unsigned int y_label_font_size() ;
svg_2d_plot & y_label_font_size(unsigned int) ;
bool y_label_on() ;
svg_2d_plot & y_label_on(bool) ;
std::string y_label_units() ;
svg_2d_plot & y_label_units(const std::string &) ;
bool y_label_units_on() ;
svg_2d_plot & y_label_units_on(bool) ;
const std::string & y_label_weight() ;
svg_2d_plot & y_label_weight(std::string) ;
double y_label_width() ;
svg_2d_plot & y_label_width(double) ;
bool y_labels_strip_e0s() ;
svg_2d_plot & y_labels_strip_e0s(bool) ;
const svg_color y_major_grid_color() ;
svg_2d_plot & y_major_grid_color(const svg_color &) ;
bool y_major_grid_on() ;
svg_2d_plot & y_major_grid_on(bool) ;
double y_major_grid_width() ;
svg_2d_plot & y_major_grid_width(double) ;
svg_2d_plot & y_major_interval(double) ;
double y_major_interval() ;
int y_major_label_rotation() ;
svg_2d_plot & y_major_label_rotation(rotate_style) ;
int y_major_labels_side() ;
svg_2d_plot & y_major_labels_side(int) ;
const svg_color y_major_tick_color() ;
svg_2d_plot & y_major_tick_color(const svg_color &) ;
svg_2d_plot & y_major_tick_length(double) ;
double y_major_tick_length() ;
double y_major_tick_width() ;
svg_2d_plot & y_major_tick_width(double) ;
double y_max() ;
double y_min() ;
const svg_color y_minor_grid_color() ;
svg_2d_plot & y_minor_grid_color(const svg_color &) ;
bool y_minor_grid_on() ;
svg_2d_plot & y_minor_grid_on(bool) ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
double y_minor_grid_width() ;
svg_2d_plot & y_minor_grid_width(double) ;
double y_minor_interval() ;
const svg_color y_minor_tick_color() ;
svg_2d_plot & y_minor_tick_color(const svg_color &) ;
double y_minor_tick_length() ;
svg_2d_plot & y_minor_tick_length(double) ;
double y_minor_tick_width() ;
svg_2d_plot & y_minor_tick_width(double) ;
unsigned int y_num_minor_ticks() ;
svg_2d_plot & y_num_minor_ticks(unsigned int) ;
const svg_color y_plusminus_color() ;
svg_2d_plot & y_plusminus_color(const svg_color &) ;
svg_2d_plot & y_plusminus_on(bool) ;
bool y_plusminus_on() ;
const std::string y_prefix() ;
std::pair< double, double > y_range() ;
svg_2d_plot & y_range(double, double) ;
const std::string y_separator() ;
const std::string y_suffix() ;
ticks_labels_style & y_ticks() ;
bool y_ticks_left_on() ;
svg_2d_plot & y_ticks_left_on(bool) ;
int y_ticks_on_window_or_axis() ;
svg_2d_plot & y_ticks_on_window_or_axis(int) ;
bool y_ticks_right_on() ;
svg_2d_plot & y_ticks_right_on(bool) ;
svg_color y_ticks_values_color() ;
svg_2d_plot & y_ticks_values_color(const svg_color &) ;
const std::string & y_ticks_values_font_family() ;
svg_2d_plot & y_ticks_values_font_family(const std::string &) ;
unsigned int y_ticks_values_font_size() ;
svg_2d_plot & y_ticks_values_font_size(unsigned int) ;
std::ios_base::fmtflags y_ticks_values_ioflags() ;
svg_2d_plot & y_ticks_values_ioflags(std::ios_base::fmtflags) ;
int y_ticks_values_precision() ;
svg_2d_plot & y_ticks_values_precision(int) ;
int y_value_ioflags() ;
svg_2d_plot & y_value_ioflags(std::ios_base::fmtflags) ;
int y_value_precision() ;
svg_2d_plot & y_value_precision(int) ;
svg_color y_values_color() ;
svg_2d_plot & y_values_color(const svg_color &) ;
const std::string & y_values_font_family() ;
svg_2d_plot & y_values_font_family(const std::string &) ;
unsigned int y_values_font_size() ;
svg_2d_plot & y_values_font_size(unsigned int) ;
std::ios_base::fmtflags y_values_ioflags() ;
svg_2d_plot & y_values_ioflags(std::ios_base::fmtflags) ;
svg_2d_plot & y_values_on(bool) ;
bool y_values_on() ;
int y_values_precision() ;
svg_2d_plot & y_values_precision(int) ;
int y_values_rotation() ;
svg_2d_plot & y_values_rotation(rotate_style) ;

// private member functions
void calculate_plot_window() ;
void draw_bars() ;
void draw_bezier_lines(const svg_2d_plot_series &) ;
void draw_histogram() ;
void draw_plot_lines() ;
void draw_plot_points() ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
  void draw_straight_lines(const svg_2d_plot_series &) ;
  void draw_y_axis() ;
  void draw_y_axis_label() ;
  void draw_y_major_tick(double, path_element &, path_element &) ;
  void draw_y_minor_tick(double, path_element &, path_element &) ;
  void set_ids() ;
  void transform_pair(std::pair< double, double > &) ;
  void update_image() ;

  // protected member functions
  void adjust_limits(double &, double &) ;
  void clear_all() ;
  void clear_background() ;
  void clear_grids() ;
  void clear_legend() ;
  void clear_plot_background() ;
  void clear_points() ;
  void clear_title() ;
  void clear_x_axis() ;
  void clear_y_axis() ;
  void draw_legend() ;
  void draw_plot_point(double, double, g_element &, plot_point_style &, unc,
                       unc) ;
  void draw_plot_point_value(double, double, g_element &, value_style &,
                             plot_point_style &, unc) ;
  void draw_plot_point_values(double, double, g_element &, g_element &,
                              const value_style &, const value_style &, unc,
                              unc) ;
  void draw_title() ;
  void draw_x_axis() ;
  void draw_x_axis_label() ;
  void draw_x_major_tick(double, path_element &, path_element &) ;
  void draw_x_minor_tick(double, path_element &, path_element &) ;
  void place_legend_box() ;
  void size_legend_box() ;
  std::string sv(double, const value_style &, bool = false) ;
  void transform_point(double &, double &) ;
  void transform_x(double &) ;
  void transform_y(double &) ;
  text_style a_style_;  // Defaults for text_style (contains font size & type etc).
  bool autoscale_check_limits_;  // true if to check autoscale values for infinity, NaN, max, min.
  double autoscale_plusminus_;  // For uncertain values, allow for plusminus ellipses showing ↵
67%, 95% and 99% confidence limits.
 < For example, if a max value is 1.2 +or- 0.02, then 1.4 will be used for autoscaling the max↵
imum.
 < Similarly, if a min value is 1.2 +or- 0.02, then 1.0 will be used for autoscaling the minimum.
  svg image_;  // Stored so as to avoid rewriting style information constantly.
  box_style image_border_;  // rectangular border box style of all image width, color...
  double legend_bottom_;  // SVG Coordinates of bottom of legend box.
  box_style legend_box_;  // rectangular box style of legend width, color...
  text_element legend_header_;  // Legend box header or title (if any).
  double legend_height_;  // Height of legend box (in pixels).
  double legend_left_;  // Left of legend box.
  bool legend_lines_;  // true if legend box should be outside the plot window.
  size_t legend_longest_;  // longest (both header & data) string in legend box.
  bool legend_on_;  // true if to provide a legend box.
  legend_places legend_place_;  // Place for any legend box.
  double legend_right_;  // SVG Coordinates of right of legend box,.
  text_style legend_style_;
  double legend_top_;  // Top of legend box.
  double legend_width_;  // Width of legend box (pixels).
  std::vector< text_element > notes_;  // Store of text for annotation.
  bool outside_legend_on_;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
  double plot_bottom_;  // SVG Y coordinate of bottom side of plot window.
  double plot_left_;  // SVG X coordinate (pixels) of left side of plot window.
  double plot_right_;  // SVG X coordinate of right side of plot window.
  double plot_top_;  // SVG Y coordinate of top side of plot window.
  box_style plot_window_border_;  // rectangular border box style of plot window width, color...
  std::string plot_window_clip_;
  bool plot_window_on_;  // true if to add a colored line for each data series in legend box.
  text_style point_symbols_style_;  // Style used for symbol marking a data point.
  std::vector< svg_2d_plot_series > serieses_;  // Store of several series of data points for ↵
transformation.
  friend svg_2d_plot_series;
  double text_margin_;  // Marginal space around text items like title, text_margin_ * font_size ↵
to get distance in svg units.
  text_element title_info_;  // Plot title text.
  bool title_on_;  // true if to display a title for the whole plot.
  text_style title_style_;  // Style for plot title.
  text_style value_style_;  // Style used for data point value label.
  double x_auto_max_value_;  // Values calculated by scale_axis (used only if x_autoscale == ↵
true).
  double x_auto_min_value_;  // Values calculated by scale_axis (used only if x_autoscale == ↵
true).
  double x_auto_tick_interval_;  // X tick major interval.
  int x_auto_ticks_;  // Number of X ticks.
  bool x_autoscale_;  // true if to use any X-axis autoscale values.
  axis_line_style x_axis_;  // Style of X axis.
  text_style x_axis_label_style_;  // Style for legend title and text.
  int x_axis_position_;  // Intersection with Y axis, or not.
  bool x_include_zero_;  // true if autoscaled, to include zero.
  text_element x_label_info_;  // X axis label text, for example: "length".
  int x_min_ticks_;  // If autoscaled, set a minimum number of X ticks.
  bool x_plusminus_on_;  // http://en.wikipedia.org/wiki/Plus-minus_sign Unicode &#0xB1; HTML ↵
&plusmn;
  double x_scale_;  // scale factor used by transform() to go from Cartesian to SVG coordinates.
  double x_shift_;  // shift factor used by transform() to go from Cartesian to SVG coordinates.
  int x_steps_;  // true if autoscale to set any prescaling to multiple of decimal 1, 2, 5, 10 ↵
etc.
  ticks_labels_style x_ticks_;  // Style of X axis tick marks and labels.
  bool x_ticks_on_;  // true if X axis to have ticks.
  double x_tight_;  // Tolerance used by autoscale to avoid extra ticks.
  text_element x_units_info_;  // X axis units, for example: "mm".
  text_element x_value_label_info_;  // X axis tick value text, for example: "1.2" or "1.2e+001".
  text_style x_value_label_style_;  // Style for data point value labels on X axis.
  bool x_values_on_;
  value_style x_values_style_;  // Data point X value marking, font, size etc.
  bool xy_values_on_;
  double y_auto_max_value_;  // Values calculated by scale_axis (used only if y_autoscale == ↵
true).
  double y_auto_min_value_;  // Values calculated by scale_axis (used only if y_autoscale == ↵
true).
  double y_auto_tick_interval_;  // tick major interval (calculated by Y autoscale).
  int y_auto_ticks_;  // Number of ticks (calculated by Y autoscale).
  bool y_autoscale_;  // true if to use any y_axis autoscale values.
  axis_line_style y_axis_;  // Style of Y axis.
  text_style y_axis_label_style_;  // Style for tick labels on Y axis.
  int y_axis_position_;  // Intersection with X axis, or not.
  bool y_include_zero_;  // true if autoscale to include zero.
  text_element y_label_info_;  // Y axis label text, for example: "volume".
  int y_min_ticks_;
  double y_scale_;  // scale factor used by transform() to go from Cartesian to SVG coordinates.
  double y_shift_;  // shift factor used by transform() to go from Cartesian to SVG coordinates.
  int y_steps_;  // If autoscaled, set any prescaling to decimal 1, 2, 5, 10 etc.
  ticks_labels_style y_ticks_;  // Style of Y axis tick marks and labels.
  bool y_ticks_on_;  // true if Y axis to have ticks.
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
  double y_tight_;  // Tolerance used by autoscale to avoid extra ticks.
  text_element y_units_info_;  // Y axis units, for example: "min". (2-D only).
  std::ios_base::fmtflags y_value_ioflags_;  // std::iosflags used for Y vale labels (default ↵
ios::dec).
  text_element y_value_label_info_;  // Y axis tick value text, for example: "1.2" or "1.2e+001".
  rotate_style y_value_label_rotation_;  // Direction point Y value labels written (default ho↵
rizontal).
  text_style y_value_label_style_;  // Style for data point value labels on Y axis.
  int y_value_precision_;  // std::ios precision used for Y value labels (default 3).
  bool y_values_on_;
  value_style y_values_style_;  // Data point Y value marking, font, size etc.
};
```

Description

See also svg_2d_plot_series to allow data values to be added. svg_2d_plot allows us to store plot state locally in svg_plot. We don't store it in "svg" because transforming the points after they are written to the document would be difficult. We store the Cartesian coordinates locally and transform them before we write them.

See also svg_1d_plot.hpp for 1-D version.

`svg_2d_plot` public construct/copy/destruct

1. ```
   svg_2d_plot();
   ```

   Default constructor, including all the very many default plot options, some of which use some or all of the style class defaults. All these settings can be changed by chainable functions, for example:

   ```
   svg_2d_plot my_plot;
   my_plot.background_color(ghostwhite) // Whole image.
     .legend_border_color(yellow) // Just the legend box.
     .legend_background_color(lightyellow)  // Fill color of the legend box.
     .plot_background_color(svg_color(white)) // Just the plot window
     .plot_border_color(svg_color(green)) // The border rectangle color.
     .plot_border_width(1) // Thin border (SVG units, default pixels).
     .title_color(red) // Title of whole image.
   ;
   ```

`svg_2d_plot` public member functions

1. ```
   svg_2d_plot & autoscale(bool b) ;
   ```

2. ```
   bool autoscale() ;
   ```

3. ```
   svg_2d_plot & autoscale_check_limits(bool b) ;
   ```

4. ```
   bool autoscale_check_limits() ;
   ```

5. ```
   svg_2d_plot & autoscale_plusminus(double) ;
   ```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

6.
```
double autoscale_plusminus() ;
```

7.
```
bool axes_on() ;
```

8.
```
svg_2d_plot & axes_on(bool is) ;
```

9.
```
svg_color background_border_color() ;
```

10.
```
svg_2d_plot & background_border_color(const svg_color & col) ;
```

11.
```
double background_border_width() ;
```

12.
```
svg_2d_plot & background_border_width(double w) ;
```

13.
```
svg_2d_plot & background_color(const svg_color & col) ;
```

14.
```
svg_color background_color() ;
```

15.
```
svg_2d_plot & boost_license_on(bool l) ;
```

16.
```
bool boost_license_on() ;
```

17.
```
int coord_precision() ;
```

18.
```
svg_2d_plot & coord_precision(int digits) ;
```

19.
```
const std::string copyright_date() ;
```

20.
```
svg_2d_plot & copyright_date(const std::string d) ;
```

21.
```
const std::string copyright_holder() ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

22.
```
svg_2d_plot & copyright_holder(const std::string d) ;
```

23.
```
double data_lines_width() ;
```

24.
```
svg_2d_plot & data_lines_width(double width) ;
```

25.
```
const std::string & description() ;
```

26.
```
svg_2d_plot & description(const std::string d) ;
```

27.
```
std::string document_title() ;
```

28.
```
svg_2d_plot & document_title(const std::string d) ;
```

29.
```
svg_2d_plot &
draw_line(double x1, double y1, double x2, double y2,
          const svg_color & col = black) ;
```

30.
```
svg_2d_plot &
draw_note(double x, double y, std::string note, rotate_style rot = horizontal,
          align_style al = center_align, const svg_color & = black,
          text_style & tsty = no_style) ;
```

31.
```
svg_2d_plot &
draw_plot_curve(double x1, double y1, double x2, double y2, double x3,
                double y3, const svg_color & col = black) ;
```

32.
```
svg_2d_plot &
draw_plot_line(double x1, double y1, double x2, double y2,
               const svg_color & col = black) ;
```

33.
```
double image_border_margin() ;
```

34.
```
svg_2d_plot & image_border_margin(double w) ;
```

35.
```
double image_border_width() ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

36.
```
svg_2d_plot & image_border_width(double w) ;
```

37.
```
std::pair< double, double > image_size() ;
```

38.
```
svg_2d_plot & image_x_size(unsigned int i) ;
```

39.
```
unsigned int image_x_size() ;
```

40.
```
svg_2d_plot & image_y_size(unsigned int i) ;
```

41.
```
unsigned int image_y_size() ;
```

42.
```
svg_color legend_background_color() ;
```

43.
```
svg_2d_plot & legend_background_color(const svg_color & col) ;
```

44.
```
svg_color legend_border_color() ;
```

45.
```
svg_2d_plot & legend_border_color(const svg_color & col) ;
```

46.
```
const std::pair< double, double > legend_bottom_right() ;
```

47.
```
bool legend_box_fill_on() ;
```

48.
```
svg_color legend_color() ;
```

49.
```
svg_2d_plot & legend_color(const svg_color & col) ;
```

50.
```
const std::string & legend_font_family() ;
```

51.
```
svg_2d_plot & legend_font_family(const std::string & family) ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

52.
```
const std::string & legend_font_weight() ;
```

53.
```
svg_2d_plot & legend_font_weight(const std::string & weight) ;
```

54.
```
int legend_header_font_size() ;
```

55.
```
svg_2d_plot & legend_header_font_size(int size) ;
```

56.
```
bool legend_lines() ;
```

57.
```
svg_2d_plot & legend_lines(bool is) ;
```

58.
```
bool legend_on() ;
```

59.
```
svg_2d_plot & legend_on(bool cmd) ;
```

60.
```
bool legend_outside() ;
```

61.
```
legend_places legend_place() ;
```

62.
```
svg_2d_plot & legend_place(legend_places l) ;
```

63.
```
const std::string legend_title() ;
```

64.
```
svg_2d_plot & legend_title(const std::string title) ;
```

65.
```
unsigned int legend_title_font_size() ;
```

66.
```
svg_2d_plot & legend_title_font_size(unsigned int size) ;
```

67.
```
const std::pair< double, double > legend_top_left() ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

68.
```
svg_2d_plot & legend_top_left(double x, double y) ;
```

69.
```
double legend_width() ;
```

70.
```
svg_2d_plot & legend_width(double width) ;
```

71.
```
svg_2d_plot &
license(std::string repro = "permits", std::string distrib = "permits",
        std::string attrib = "requires", std::string commercial = "permits",
        std::string derivative = "permits") ;
```

72.
```
const std::string license_attribution() ;
```

73.
```
const std::string license_commercialuse() ;
```

74.
```
const std::string license_distribution() ;
```

75.
```
svg_2d_plot & license_on(bool l) ;
```

76.
```
bool license_on() ;
```

77.
```
const std::string license_reproduction() ;
```

78.
```
svg_color limit_color() ;
```

79.
```
svg_2d_plot & limit_color(const svg_color &) ;
```

80.
```
svg_color limit_fill_color() ;
```

81.
```
svg_2d_plot & limit_fill_color(const svg_color &) ;
```

82.
```
svg_color one_sd_color() ;
```

83.
```
svg_2d_plot & one_sd_color(const svg_color &) ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

84.
```
template<class T , class U >
  svg_2d_plot_series &
  plot(const T & begin, const T & end, const std::string & title = "",
       U functor = pair_double_2d_convert) ;
```

Returns: Reference to data series just added to make chainable.

85.
```
template<class T >
  svg_2d_plot_series &
  plot(const T & begin, const T & end, const std::string & title = "") ;
```

Add a data series to the plot (by default, converting automatically to unc doubles). This version permits part of the container to be used, a partial range, using iterators begin to end. For example:

```
my_2d_plot.plot(my_data.begin(), my_data.end(), "My container");
```

```
my_2d_plot.plot(&my_data[1], &my_data[], "my_data 1 to 3"); // Add part of data series.
```

Returns: Reference to data series just added to make chainable.

86.
```
template<class T , class U >
  svg_2d_plot_series &
  plot(const T & container, const std::string & title = "",
       U functor = pair_double_2d_convert) ;
```

This version permits a custom functor (rather than default conversion to double). Note that this version assumes that *ALL* the data value in the container is used.

Returns: Reference to data series just added to make chainable.

87.
```
template<class T >
  svg_2d_plot_series &
  plot(const T & container, const std::string & title = "") ;
```

Add a container of a data series to the plot. This version assumes that **ALL** the data values in the container are used.

```
my_plot.plot(data1, "Sqrt(x)");
```

Version converting to double with `pair_double_2d_convert`.

Returns: Reference to data series just added to make chainable.

88.
```
svg_color plot_background_color() ;
```

89.
```
svg_2d_plot & plot_background_color(const svg_color & col) ;
```

90.
```
svg_color plot_border_color() ;
```

91.
```
svg_2d_plot & plot_border_color(const svg_color & col) ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

92.
```
svg_2d_plot & plot_border_width(double w) ;
```

93.
```
double plot_border_width() ;
```

94.
```
bool plot_window_on() ;
```

95.
```
svg_2d_plot & plot_window_on(bool cmd) ;
```

96.
```
std::pair< double, double > plot_window_x() ;
```

97.
```
svg_2d_plot & plot_window_x(double min_x, double max_x) ;
```

98.
```
double plot_window_x_left() ;
```

99.
```
double plot_window_x_right() ;
```

100.
```
std::pair< double, double > plot_window_y() ;
```

101.
```
svg_2d_plot & plot_window_y(double min_y, double max_y) ;
```

102.
```
double plot_window_y_bottom() ;
```

103.
```
double plot_window_y_top() ;
```

104.
```
svg_2d_plot & size(unsigned int x, unsigned int y) ;
```

105.
```
svg_color three_sd_color() ;
```

106.
```
svg_2d_plot & three_sd_color(const svg_color &) ;
```

107.
```
const std::string title() ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
svg_2d_plot & title(const std::string title) ;
```

```
svg_color title_color() ;
```

```
svg_2d_plot & title_color(const svg_color & col) ;
```

```
align_style title_font_alignment() ;
```

```
svg_2d_plot & title_font_alignment(align_style alignment) ;
```

```
const std::string & title_font_decoration() ;
```

```
svg_2d_plot & title_font_decoration(const std::string & decoration) ;
```

```
const std::string & title_font_family() ;
```

```
svg_2d_plot & title_font_family(const std::string & family) ;
```

```
int title_font_rotation() ;
```

```
svg_2d_plot & title_font_rotation(rotate_style rotate) ;
```

```
unsigned int title_font_size() ;
```

```
svg_2d_plot & title_font_size(unsigned int i) ;
```

```
const std::string & title_font_stretch() ;
```

```
svg_2d_plot & title_font_stretch(const std::string & stretch) ;
```

```
const std::string & title_font_style() ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

124
```
svg_2d_plot & title_font_style(const std::string & style) ;
```

125
```
const std::string & title_font_weight() ;
```

126
```
svg_2d_plot & title_font_weight(const std::string & weight) ;
```

127.
```
bool title_on() ;
```

128
```
svg_2d_plot & title_on(bool cmd) ;
```

129
```
svg_color two_sd_color() ;
```

130
```
svg_2d_plot & two_sd_color(const svg_color &) ;
```

131
```
svg_2d_plot & write(std::ostream & s_out) ;
```

Write the SVG image to an ostream.

Returns: reference to svg_2d_plot to make chainable.

132
```
svg_2d_plot & write(const std::string & file) ;
```

Write the plot image to a named file (default suffix .svg).

Returns: reference to svg_2d_plot_series to make chainable.
Returns: reference to svg_2d_plot to make chainable.

133
```
double x_auto_max_value() ;
```

134
```
double x_auto_min_value() ;
```

135
```
double x_auto_tick_interval() ;
```

136
```
int x_auto_ticks() ;
```

137.
```
svg_2d_plot & x_autoscale(const T & begin, const T & end) ;
```

autoscale X axis using the whole data series.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
svg_2d_plot & x_autoscale(const T & container) ;
```

autoscale X axis using a pair of doubles.

```
svg_2d_plot & x_autoscale(std::pair< double, double > p) ;
```

```
svg_2d_plot & x_autoscale(bool b) ;
```

```
bool x_autoscale() ;
```

```
axis_line_style & x_axis() ;
```

Returns: true if horizontal X axis line to be drawn.

```
svg_color x_axis_color() ;
```

```
svg_2d_plot & x_axis_color(const svg_color & col) ;
```

```
svg_color x_axis_label_color() ;
```

```
svg_2d_plot & x_axis_label_color(const svg_color & col) ;
```

```
bool x_axis_on() ;
```

```
svg_2d_plot & x_axis_on(bool is) ;
```

```
const std::string x_axis_position() ;
```

```
double x_axis_width() ;
```

```
svg_2d_plot & x_axis_width(double width) ;
```

```
svg_2d_plot &
x_decor(const std::string & pre, const std::string & sep = "",
        const std::string & suf = "") ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
svg_color x_df_color() ;
```

```
svg_2d_plot & x_df_color(const svg_color & col) ;
```

```
bool x_df_on() ;
```

```
svg_2d_plot & x_df_on(bool b) ;
```

```
std::string x_label() ;
```

```
svg_2d_plot & x_label(const std::string & str) ;
```

```
svg_color x_label_color() ;
```

```
svg_2d_plot & x_label_color(const svg_color & col) ;
```

```
const std::string & x_label_font_family() ;
```

```
svg_2d_plot & x_label_font_family(const std::string & family) ;
```

```
unsigned int x_label_font_size() ;
```

```
svg_2d_plot & x_label_font_size(unsigned int i) ;
```

```
bool x_label_on() ;
```

Returns: true if to include an X axis text label.

```
svg_2d_plot & x_label_on(bool cmd) ;
```

Set to include an X axis text label.

x_label_units
Returns: reference to svg_2d_plot to make chainable.

```
std::string x_label_units() ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
svg_2d_plot & x_label_units(const std::string & str) ;
```

```
bool x_label_units_on() ;
```

```
svg_2d_plot & x_label_units_on(bool cmd) ;
```

```
double x_label_width() ;
```

```
svg_2d_plot & x_label_width(double width) ;
```

```
svg_2d_plot & x_labels_strip_e0s(bool cmd) ;
```

```
svg_color x_major_grid_color() ;
```

```
svg_2d_plot & x_major_grid_color(const svg_color & col) ;
```

```
bool x_major_grid_on() ;
```

```
svg_2d_plot & x_major_grid_on(bool is) ;
```

```
double x_major_grid_width() ;
```

```
svg_2d_plot & x_major_grid_width(double w) ;
```

```
double x_major_interval() ;
```

```
svg_2d_plot & x_major_interval(double inter) ;
```

```
rotate_style x_major_label_rotation() ;
```

```
svg_2d_plot & x_major_label_rotation(rotate_style rot) ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
int x_major_labels_side() ;
```

Returns: Label values side for major ticks.

```
svg_2d_plot & x_major_labels_side(int) ;
```

Side for major ticks label values:
Returns: reference to svg_2d_plot to make chainable.

```
double x_major_tick() ;
```

```
svg_2d_plot & x_major_tick(double d) ;
```

```
svg_color x_major_tick_color() ;
```

```
svg_2d_plot & x_major_tick_color(const svg_color & col) ;
```

```
double x_major_tick_length() ;
```

```
svg_2d_plot & x_major_tick_length(double length) ;
```

```
double x_major_tick_width() ;
```

```
svg_2d_plot & x_major_tick_width(double width) ;
```

```
double x_max() ;
```

```
svg_2d_plot & x_max(double x) ;
```

```
double x_min() ;
```

```
svg_2d_plot & x_min(double min_x) ;
```

```
int x_min_ticks() ;
```

```
svg_2d_plot & x_min_ticks(int min_ticks) ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
svg_color x_minor_grid_color() ;
```

```
svg_2d_plot & x_minor_grid_color(const svg_color & col) ;
```

```
bool x_minor_grid_on() ;
```

```
svg_2d_plot & x_minor_grid_on(bool is) ;
```

```
double x_minor_grid_width() ;
```

```
svg_2d_plot & x_minor_grid_width(double w) ;
```

```
svg_2d_plot & x_minor_interval(double interval) ;
```

```
double x_minor_interval() ;
```

```
svg_color x_minor_tick_color() ;
```

```
svg_2d_plot & x_minor_tick_color(const svg_color & col) ;
```

```
double x_minor_tick_length() ;
```

```
svg_2d_plot & x_minor_tick_length(double length) ;
```

```
double x_minor_tick_width() ;
```

```
svg_2d_plot & x_minor_tick_width(double width) ;
```

```
unsigned int x_num_minor_ticks() ;
```

```
svg_2d_plot & x_num_minor_ticks(unsigned int num) ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
svg_color x_plusminus_color() ;
```

```
svg_2d_plot & x_plusminus_color(const svg_color & col) ;
```

```
bool x_plusminus_on() ;
```

```
svg_2d_plot & x_plusminus_on(bool b) ;
```

```
const std::string x_prefix() ;
```

```
std::pair< double, double > x_range() ;
```

```
svg_2d_plot & x_range(double min_x, double max_x) ;
```

```
const std::string x_separator() ;
```

```
int x_steps() ;
```

```
svg_2d_plot & x_steps(int steps) ;
```

```
const std::string x_suffix() ;
```

```
ticks_labels_style & x_ticks() ;
```

Returns: true if ticks are to be marked on the X axis.

```
bool x_ticks_down_on() ;
```

```
svg_2d_plot & x_ticks_down_on(bool cmd) ;
```

```
int x_ticks_on_window_or_axis() ;
```

Returns: if ticks on the plot window or on the X axis.

```
svg_2d_plot & x_ticks_on_window_or_axis(int) ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Set if ticks on the plot window or on the X axis.
Returns: reference to svg_2d_plot to make chainable.

```
bool x_ticks_up_on() ;
```

```
svg_2d_plot & x_ticks_up_on(bool cmd) ;
```

```
svg_color x_ticks_values_color() ;
```

```
svg_2d_plot & x_ticks_values_color(const svg_color & col) ;
```

```
const std::string & x_ticks_values_font_family() ;
```

```
svg_2d_plot & x_ticks_values_font_family(const std::string & family) ;
```

```
unsigned int x_ticks_values_font_size() ;
```

```
svg_2d_plot & x_ticks_values_font_size(unsigned int i) ;
```

```
std::ios_base::fmtflags x_ticks_values_ioflags() ;
```

```
svg_2d_plot & x_ticks_values_ioflags(std::ios_base::fmtflags f) ;
```

```
int x_ticks_values_precision() ;
```

```
svg_2d_plot & x_ticks_values_precision(int p) ;
```

```
double x_tight() ;
```

```
svg_2d_plot & x_tight(double tight) ;
```

```
unsigned int x_value_font_size() ;
```

```
svg_2d_plot & x_value_font_size(unsigned int i) ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
int x_value_ioflags() ;
```

```
svg_2d_plot & x_value_ioflags(int flags) ;
```

```
int x_value_precision() ;
```

```
svg_2d_plot & x_value_precision(int digits) ;
```

```
svg_color x_values_color() ;
```

```
svg_2d_plot & x_values_color(const svg_color & col) ;
```

```
const std::string & x_values_font_family() ;
```

```
svg_2d_plot & x_values_font_family(const std::string & family) ;
```

```
unsigned int x_values_font_size() ;
```

```
svg_2d_plot & x_values_font_size(unsigned int i) ;
```

```
std::ios_base::fmtflags x_values_ioflags() ;
```

```
svg_2d_plot & x_values_ioflags(std::ios_base::fmtflags f) ;
```

```
bool x_values_on() ;
```

```
svg_2d_plot & x_values_on(bool b) ;
```

```
int x_values_precision() ;
```

```
svg_2d_plot & x_values_precision(int p) ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

264
```
int x_values_rotation() ;
```

265
```
svg_2d_plot & x_values_rotation(rotate_style rotate) ;
```

266
```
bool x_with_zero() ;
```

267
```
svg_2d_plot & x_with_zero(bool b) ;
```

268
```
bool xy_autoscale() ;
```

Returns: true if to autoscale both X and Y Axes.

269
```
template<class T > svg_2d_plot & xy_autoscale(const T & container) ;
```

Whole data series to use to calculate autoscaled values for **both** X and Y axes.

Returns: reference to svg_2d_plot to make chainable.

270
```
svg_2d_plot & xy_values_on(bool b) ;
```

Set true if values of X and Y data points are shown (as 1.23). (Will override x_values_on and/or y_values_on that would otherwise cause overwriting).

Returns: reference to svg_2d_plot to make chainable.

271
```
bool xy_values_on() ;
```

Returns: true if values of X and Y data points are shown (for example: 1.23).

272
```
template<class T > svg_2d_plot & y_autoscale(const T & container) ;
```

Whole data series to use to calculate autoscaled values.

273
```
template<class T > svg_2d_plot & y_autoscale(const T & begin, const T & end) ;
```

Data series using iterator's range to use to calculate autoscaled values.

Returns: reference to svg_2d_plot to make chainable.

274
```
svg_2d_plot & y_autoscale(std::pair< double, double > p) ;
```

Set Y min & max values as a **pair** to use to autoscale.

Returns: reference to svg_2d_plot to make chainable.

275
```
svg_2d_plot & y_autoscale(double first, double second) ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Set minimum & maximum Y values to use to autoscale Y axis.

Returns: reference to svg_2d_plot to make chainable.

```
svg_2d_plot & y_autoscale(bool b) ;
```

Set true if to autoscale minimum and maximum for Y axis.

Returns: reference to svg_2d_plot to make chainable.

```
bool y_autoscale() ;
```

Returns: true if to autoscale minimum and maximum for Y axis.

```
axis_line_style & y_axis() ;
```

Returns: true if vertical Y axis line to be drawn.

```
svg_color y_axis_color() ;
```

Returns: the stroke color.

```
svg_2d_plot & y_axis_color(const svg_color & col) ;
```

Set Y axis linecolor. (set only stroke color).

Returns: reference to svg_2d_plot to make chainable.

```
svg_color y_axis_label_color() ;
```

Returns: the y axis label stroke color.

```
svg_2d_plot & y_axis_label_color(const svg_color & col) ;
```

Set stroke color.

Returns: reference to svg_2d_plot to make chainable.

```
bool y_axis_on() ;
```

```
svg_2d_plot & y_axis_on(bool is) ;
```

```
const std::string y_axis_position() ;
```

Returns: text information about Y axis position.

```
svg_color y_axis_value_color() ;
```

Returns: color of Y-axis tick value labels.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

287.
```
svg_2d_plot & y_axis_value_color(const svg_color & col) ;
```

Set color of Y axis value labels.

Returns: reference to svg_2d_plot to make chainable.

288
```
double y_axis_width() ;
```

Returns: width of Y axis line.

289
```
svg_2d_plot & y_axis_width(double width) ;
```

Set width of Y axis line.

Returns: reference to svg_2d_plot to make chainable.

290
```
svg_2d_plot &
y_decor(const std::string & pre, const std::string & sep = "",
        const std::string & suf = "") ;
```

Set prefix, separator and suffix for Note if you want a space, you must use a Unicode space "\ ", for example, ",\ " rather than ", ".

Returns: reference to svg_2d_plot to make chainable.

291.
```
const svg_color y_df_color() ;
```

Returns: color of Y degrees of freedom.

292.
```
svg_2d_plot & y_df_color(const svg_color & col) ;
```

Set color of Y degrees of freedom.

Returns: reference to svg_2d_plot to make chainable.

293
```
svg_2d_plot & y_df_on(bool b) ;
```

Set true if values of Y data points are to include degrees of freedom estimates.

Returns: reference to svg_2d_plot to make chainable.

294
```
bool y_df_on() ;
```

Returns: true if values of Y data points are to include degrees of freedom estimates.

295
```
std::string y_label() ;
```

296
```
svg_2d_plot & y_label(const std::string & str) ;
```

297.
```
std::string y_label_axis() ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Returns: text to label Y axis.

```
svg_2d_plot & y_label_axis(const std::string & str) ;
```

Set text to label Y-axis.

Returns: reference to svg_2d_plot to make chainable.

```
svg_color y_label_color() ;
```

```
svg_2d_plot & y_label_color(const svg_color & col) ;
```

```
const std::string & y_label_font_family() ;
```

Returns: the font family for label on Y axis.

```
svg_2d_plot & y_label_font_family(const std::string & family) ;
```

Set Y axis label text font family (for example: "Lucida console sans").

Returns: reference to svg_2d_plot to make chainable.

```
unsigned int y_label_font_size() ;
```

Returns: Y axis label text font size.

```
svg_2d_plot & y_label_font_size(unsigned int i) ;
```

Set Y axis label text font size.

```
bool y_label_on() ;
```

Returns: true if Y label is on.

```
svg_2d_plot & y_label_on(bool cmd) ;
```

Returns: reference to svg_2d_plot to make chainable.

```
std::string y_label_units() ;
```

```
svg_2d_plot & y_label_units(const std::string & str) ;
```

```
bool y_label_units_on() ;
```

Returns: true if to add units text to the Y axis label.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

310
```
svg_2d_plot & y_label_units_on(bool b) ;
```

Set true to add units text to the Y axis label.

Returns: reference to svg_2d_plot to make chainable.

311
```
const std::string & y_label_weight() ;
```

Returns: Y axis label text font weight (for example: "bold").

312
```
svg_2d_plot & y_label_weight(std::string s) ;
```

Set Y axis label text font weight (for example: "bold"). ("bold" is only one that works so far, and quality may be poor for some browsers).

Returns: reference to svg_2d_plot to make chainable.

313
```
double y_label_width() ;
```

Returns: color of Y axis value labels.

314
```
svg_2d_plot & y_label_width(double width) ;
```

Set width of Y axis value labels.

Returns: reference to svg_2d_plot to make chainable.

315
```
bool y_labels_strip_e0s() ;
```

Returns: true if to strip unnecessary zeros, signs from labels.

316
```
svg_2d_plot & y_labels_strip_e0s(bool cmd) ;
```

If true then strip unnecessary zeros, signs from labels.

Returns: reference to svg_2d_plot to make chainable.

317
```
const svg_color y_major_grid_color() ;
```

Returns: color of Y major grid lines.

318
```
svg_2d_plot & y_major_grid_color(const svg_color & col) ;
```

Set color of Y major grid lines.

Returns: reference to svg_2d_plot to make chainable.

319
```
bool y_major_grid_on() ;
```

Returns: true to include major grid lines.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

320
```
svg_2d_plot & y_major_grid_on(bool is) ;
```

Set true to include major grid lines.

Returns: reference to svg_2d_plot to make chainable.

321
```
double y_major_grid_width() ;
```

Set width of major grid lines.

322
```
svg_2d_plot & y_major_grid_width(double width) ;
```

Set width of major grid lines.

Returns: reference to svg_2d_plot to make chainable.

323
```
svg_2d_plot & y_major_interval(double inter) ;
```

Set major interval between ticks on Y axis.

Returns: reference to svg_2d_plot to make chainable.

324
```
double y_major_interval() ;
```

Returns: major interval between ticks on Y axis.

325
```
int y_major_label_rotation() ;
```

Returns: rotation of Y axis major tick labels.

326
```
svg_2d_plot & y_major_label_rotation(rotate_style rot) ;
```

Rotation of labels for major ticks on vertical Y axis line.

- `rot` Default is horizontal.

  rotate_style for possible values: horizontal, uphill...

327
```
int y_major_labels_side() ;
```

Returns: Position of labels (if any) for major ticks on vertical Y axis line.

328
```
svg_2d_plot & y_major_labels_side(int place) ;
```

Position of labels for major ticks on vertical Y axis line.

- `side` > 0 label to left of Y axis line (default),
- `side` = 0 (false) means no major tick labels on Y axis.
- `side` > 0 means to right of Y axis line.

Returns: reference to svg_2d_plot to make chainable.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

329
```
const svg_color y_major_tick_color() ;
```

Returns: color of Y major tick lines.

330
```
svg_2d_plot & y_major_tick_color(const svg_color & col) ;
```

Set color of Y major tick lines.

Returns: reference to svg_2d_plot to make chainable.

331
```
svg_2d_plot & y_major_tick_length(double length) ;
```

Set major tick length on Y axis.

Returns: reference to svg_2d_plot to make chainable.

332
```
double y_major_tick_length() ;
```

Returns: major tick length on Y axis.

333
```
double y_major_tick_width() ;
```

Returns: width of major ticks on Y axis.

334
```
svg_2d_plot & y_major_tick_width(double width) ;
```

Set width of major ticks on Y axis.

Returns: reference to svg_2d_plot to make chainable.

335
```
double y_max() ;
```

Returns: minimum for Y axis.

336
```
double y_min() ;
```

Returns: minimum for Y axis.

337
```
const svg_color y_minor_grid_color() ;
```

Returns: color of Y minor grid lines.

338
```
svg_2d_plot & y_minor_grid_color(const svg_color & col) ;
```

Set color of Y minor grid lines.

Returns: reference to svg_2d_plot to make chainable.

339
```
bool y_minor_grid_on() ;
```

Set true to include minor grid lines.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
svg_2d_plot & y_minor_grid_on(bool is) ;
```

Set true to include minor grid lines.

Returns: reference to svg_2d_plot to make chainable.

```
double y_minor_grid_width() ;
```

Returns: width of minor grid lines.

```
svg_2d_plot & y_minor_grid_width(double width) ;
```

Set width of minor grid lines.

Returns: reference to svg_2d_plot to make chainable.

```
double y_minor_interval() ;
```

```
const svg_color y_minor_tick_color() ;
```

Returns: color of Y minor tick lines.

```
svg_2d_plot & y_minor_tick_color(const svg_color & col) ;
```

Set color of Y minor tick lines.

Returns: reference to svg_2d_plot to make chainable.

```
double y_minor_tick_length() ;
```

Returns: minor tick length on Y axis.

```
svg_2d_plot & y_minor_tick_length(double length) ;
```

Set minor tick length on Y axis.

Returns: reference to svg_2d_plot to make chainable.

```
double y_minor_tick_width() ;
```

Returns: width of minor ticks on Y axis.

```
svg_2d_plot & y_minor_tick_width(double width) ;
```

Set width of minor ticks on Y axis.

Returns: reference to svg_2d_plot to make chainable.

```
unsigned int y_num_minor_ticks() ;
```

Returns: number of minor ticks on Y axis.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

351.
```
svg_2d_plot & y_num_minor_ticks(unsigned int num) ;
```

Set number of minor ticks on Y axis.

Returns: reference to svg_2d_plot to make chainable.

352.
```
const svg_color y_plusminus_color() ;
```

Returns: color of Y uncertainty of value.

353.
```
svg_2d_plot & y_plusminus_color(const svg_color & col) ;
```

Set color of Y uncertainty of value.

Returns: reference to svg_2d_plot to make chainable.
Returns: reference to svg_2d_plot to make chainable.

354.
```
svg_2d_plot & y_plusminus_on(bool b) ;
```

Set true if values of Y data points are to include uncertainty estimates.

355.
```
bool y_plusminus_on() ;
```

Returns: true if values of Y data points are to include uncertainty estimates.

356.
```
const std::string y_prefix() ;
```

Get the prefix (only used if separator != "")

357.
```
std::pair< double, double > y_range() ;
```

Returns: the range (max and min) for Y axis.

358.
```
svg_2d_plot & y_range(double min_y, double max_y) ;
```

Set the range (max and min) for Y axis from the parameters provided.

Returns: reference to svg_2d_plot to make chainable.

359.
```
const std::string y_separator() ;
```

Get separator (also controls use of the prefix & suffix - they are only used if separator != ""). Note For a space, you must use a Unicode space "\ ", for example, ",\ " rather than ", ".

360.
```
const std::string y_suffix() ;
```

Get the suffix (only used if separator != "")

361.
```
ticks_labels_style & y_ticks() ;
```

Returns: true if ticks are to be marked on the Y axis.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
bool y_ticks_left_on() ;
```

Returns: true if ticks on the Y axis are to be on left of axis line.

```
svg_2d_plot & y_ticks_left_on(bool cmd) ;
```

Set true if ticks on the Y axis are to be on left of axis line.

Returns: reference to svg_2d_plot to make chainable.

```
int y_ticks_on_window_or_axis() ;
```

Returns: true if Y axis ticks wanted on the window (rather than on axis). -1 left of plot window, 0 on Y axis, +1 right of plot window.

```
svg_2d_plot & y_ticks_on_window_or_axis(int) ;
```

Set Y ticks on window or axis

- cmd -1 left of plot window,
- cmd 0 on Y axis.
- cmd +1 right of plot window.

Returns: reference to svg_2d_plot to make chainable.

```
bool y_ticks_right_on() ;
```

Returns: true if ticks on the Y axis are to be on right of axis line.

```
svg_2d_plot & y_ticks_right_on(bool cmd) ;
```

Set true if ticks on the Y axis are to be on right of axis line.

Returns: reference to svg_2d_plot to make chainable.

```
svg_color y_ticks_values_color() ;
```

Returns: color for Y-axis ticks values.

```
svg_2d_plot & y_ticks_values_color(const svg_color & col) ;
```

Set color for Y_axis tick values.

Returns: reference to svg_2d_plot to make chainable.

```
const std::string & y_ticks_values_font_family() ;
```

Returns: font family for Y axis ticks values.

```
svg_2d_plot & y_ticks_values_font_family(const std::string & family) ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Set font family for Y axis ticks values.

Returns: reference to svg_2d_plot to make chainable.

```
unsigned int y_ticks_values_font_size() ;
```

Returns: font size for Y axis values.

```
svg_2d_plot & y_ticks_values_font_size(unsigned int i) ;
```

Set font size for Y axis ticks values (svg units, default pixels).

Returns: reference to svg_2d_plot to make chainable.

```
std::ios_base::fmtflags y_ticks_values_ioflags() ;
```

Returns: iostream format flags of ticks Y values. Might be used to set hexadecimal, fixed and scientific, (std::ios::scientific).

```
svg_2d_plot & y_ticks_values_ioflags(std::ios_base::fmtflags f) ;
```

Set iostream format flags of ticks Y values. Useful to set hexadecimal, fixed and scientific, (std::ios::scientific).

```
int y_ticks_values_precision() ;
```

Returns: iostream decimal digits precision of ticks Y values..

```
svg_2d_plot & y_ticks_values_precision(int p) ;
```

Set iostream decimal digits precision of ticks Y values.

```
int y_value_ioflags() ;
```

Returns: All stream ioflags for control of format of Y tick value labels.

```
svg_2d_plot & y_value_ioflags(std::ios_base::fmtflags flags) ;
```

set IO flags of Y tick label values (default 0X201 == dec).

```
my_plot.x_value_ioflags(ios::dec | ios::scientific).x_value_precision(2);
```

Returns: reference to svg_2d_plot to make chainable.

```
int y_value_precision() ;
```

Returns: precision of Y tick value labels in decimal digits (default 3).

```
svg_2d_plot & y_value_precision(int digits) ;
```

Set precision of Y tick label values in decimal digits (default 3).

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
my_plot.x_value_ioflags(ios::dec | ios::scientific).x_value_precision(2);
```

Returns: reference to svg_2d_plot to make chainable.

```
svg_color y_values_color() ;
```

Returns: color for Y axis values.

```
svg_2d_plot & y_values_color(const svg_color & col) ;
```

Set color for Y axis values.

Returns: reference to svg_2d_plot to make chainable.

```
const std::string & y_values_font_family() ;
```

Returns: font family for Y axis values.

```
svg_2d_plot & y_values_font_family(const std::string & family) ;
```

Set font family for Y axis values.

Returns: reference to svg_2d_plot to make chainable.

```
unsigned int y_values_font_size() ;
```

Returns: font size for Y axis values.

```
svg_2d_plot & y_values_font_size(unsigned int i) ;
```

Set font size for Y axis values.

Returns: reference to svg_2d_plot to make chainable.

```
std::ios_base::fmtflags y_values_ioflags() ;
```

Returns: iostream format flags for data point values.

```
svg_2d_plot & y_values_ioflags(std::ios_base::fmtflags f) ;
```

Set iostream format flags for data point values.

Returns: reference to svg_2d_plot to make chainable.

```
svg_2d_plot & y_values_on(bool b) ;
```

Set true if values of Y data points are shown (for example: 1.23).

Returns: reference to svg_2d_plot to make chainable.

```
bool y_values_on() ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Returns: true if values of Y data points are shown (for example: 1.23). (Will override xy_values_on that would otherwise cause overwriting). So the last values_on setting will prevail.

```
int y_values_precision() ;
```

Returns: iostream precision for data points Y values.

```
svg_2d_plot & y_values_precision(int p) ;
```

Set iostream precision for data points Y values.

Returns: reference to svg_2d_plot to make chainable.

```
int y_values_rotation() ;
```

Returns: rotation for value labels on Y axis.

```
svg_2d_plot & y_values_rotation(rotate_style rotate) ;
```

Set rotation for value labels on Y axis ticks.

rotate_style
Returns: reference to svg_2d_plot to make chainable.

`svg_2d_plot` private member functions

1.
```
void calculate_plot_window() ;
```

The plot window is used to set a clip path: this ensures that data points and lines (and anything else) outside this window are NOT drawn.

All calculation use svg units, pixels by default.

2.
```
void draw_bars() ;
```

Draw normal bar chart for 'good' non-limit points.

3.
```
void draw_bezier_lines(const svg_2d_plot_series & series) ;
```

Add Bezier curve line between data points.

4.
```
void draw_histogram() ;
```

Draw a histogram with variable width but contiguous bins. Histograms differ from bar charts in the the *area* denotes the value, whereas the bar *height* denotes the value for a bar chart. bin widths are provided from the X-axis data series values. The 1st data X-value provides the start of the 1st bin, the 2nd data X-value provides the end of the 1st bin, and the 1st Y-value the area of the 1st bin, and the start of the second bin, and so on, until the width of last bin is calculated from the last data point in series, that must have a zero area. ? NaN Bins can be the same (most common) or different widths. Intervals must not overlap and bins must be adjacent. http://en.wikipedia.org/wiki/Histogram

Attempts to allow a row or horizontal were abandoned because of complications with the use of map which orders the x values providing the bins. Using the y values for the bins implies changing the Y axes labeling and scaling too.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

5.
```
void draw_plot_lines() ;
```

Draw line through data series, Bezier curved or straight, or none.

6.
```
void draw_plot_points() ;
```

Draw normal 'good' non-limit points, and then any 'at limits' points.

7.
```
void draw_straight_lines(const svg_2d_plot_series & series) ;
```

Add line between series of data points (straight rather than a Bezier curve).

8.
```
void draw_y_axis() ;
```

Draw the Y axis line, grids and ticks with labels.

9.
```
void draw_y_axis_label() ;
```

Draw a vertical Y-axis label, and optional y units.

10.
```
void draw_y_major_tick(double value, path_element & tick_path,
                       path_element & grid_path) ;
```

Draw a Y axis major tick, tick value labels & grids.

11.
```
void draw_y_minor_tick(double value, path_element & tick_path,
                       path_element & grid_path) ;
```

Draw a Y-axis minor tick and optional grid. (minor ticks do not have value labels).

12.
```
void set_ids() ;
```

document ids for use in <g id = "PLOT_TITLE".../>

13.
```
void transform_pair(std::pair< double, double > & pt) ;
```

Transform both x and y from Cartesian to SVG coordinates. SVG image is 0, 0 at top left, Cartesian at bottom left.

14.
```
void update_image() ;
```

Draw the whole SVG image.

`svg_2d_plot` protected member functions

1.
```
void adjust_limits(double & x, double & y) ;
```

If value of a data point reaches limit of max, min, infinity, use the appropriate plot min or max value instead.

2.
```
void clear_all() ;
```

Clear all layers of the plot.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

When writing to multiple documents, the contents of the plot may change significantly between. Rather than figuring out what has and has not changed, just erase the contents of the legend, title... in the document and start over.

3.
```
void clear_background() ;
```

Clear the whole image background layer of the SVG plot.

4.
```
void clear_grids() ;
```

Clear the grids layer of the SVG plot.

5.
```
void clear_legend() ;
```

Clear the legend layer of the SVG plot.

6.
```
void clear_plot_background() ;
```

Clear the plot area background layer of the SVG plot.

7.
```
void clear_points() ;
```

Clear the data points layer of the SVG plot.

8.
```
void clear_title() ;
```

Clear the plot title layer of the SVG plot.

9.
```
void clear_x_axis() ;
```

Clear the X axis layer of the SVG plot.

10.
```
void clear_y_axis() ;
```

Clear the Y axis layer of the SVG plot.

11.
```
void draw_legend() ;
```

Draw the legend border, text header (if any) and marker lines and/or shapes.

12.
```
void draw_plot_point(double x, double y, g_element & g_ptr,
                     plot_point_style & sty, unc ux, unc uy) ;
```

Draw a plot data point marker shape whose size and stroke and fill colors are specified in plot_point_style.

13.
```
void draw_plot_point_value(double x, double y, g_element & g_ptr,
                           value_style & val_style,
                           plot_point_style & point_style, unc uvalue) ;
```

Write one data point (X or Y) value as a string, for example "1.23e-2", near the data point marker. Unecessary e, +, & leading exponent zeros may optionally be stripped, and the position and rotation controlled. Uncertainty estimate, typically standard deviation (half conventional 95% confidence 'plus or minus') may be optionally be appended. Degrees of freedom estimate (number

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

of replicates) may optionally be appended. For example: "3.45 +-0.1(10)" The precision and format (scientific, fixed), and color and font type and size can be controlled too.

Unicode space plusminus glyph.

14.
```
void draw_plot_point_values(double x, double y, g_element & x_g_ptr,
                            g_element & y_g_ptr, const value_style & x_sty,
                            const value_style & y_sty, unc uncx, unc uncy) ;
```

Write the **pair** of data point's X and Y values as a string. If a separator, then both on the same line, for example "1.23, 3.45", or "[5.6, 7.8] X value_style is used to provide the prefix and separator, and Y value_style to provide the suffix. For example, x_style prefix("[X=", and separator ",\<ampersand\>\#x00A0;Y= ", " and Y value_style = "]" will produce a value label like "[X=-1.23, Y=4.56]" Note the need to use a Unicode space <ampersand>#x00A0; for get space for all browsers. For as long a string as this you may need to make the total image size bigger, and to orient the value labels with care. draw_plot_point_values is only when both x and Y pairs are wanted.

Also strip unnecessary e, + and leading exponent zeros, if required.

Unicode space plusminus glyph.

15.
```
void draw_title() ;
```

Draw title (for the whole plot). Update title_info_ with position. Assumes align = center_align Using center_align will ensure that title will center correctly because the render engine does the centering. (Even if the original string is made much longer because it contains Unicode, Greek, math symbols etc, taking about 8 characters per symbol. For example, the Unicode symbol for square root is "√" but takes only about one character width).

16.
```
void draw_x_axis() ;
```

Draw horizontal X-axis line & plot window line to hold, and ticks and grids.

17.
```
void draw_x_axis_label() ;
```

Draw the X-axis label text (for example, length), and append any optional units (for example, km).

18.
```
void draw_x_major_tick(double value, path_element & tick_path,
                       path_element & grid_path) ;
```

Draw major ticks - and grid too if wanted. If major_value_labels_side then value shown beside the major tick.

19.
```
void draw_x_minor_tick(double value, path_element & tick_path,
                       path_element & grid_path) ;
```

Draw X-axis minor ticks, and optional grid. (Value is NOT (yet) shown beside the minor tick).

20.
```
void place_legend_box() ;
```

Place legend box (if required).

21.
```
void size_legend_box() ;
```

Calculate how big the legend box needs to be.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

22.
```
std::string sv(double v, const value_style & sty, bool precise = false) ;
```

Strip from double value any unecessary e, +, & leading exponent zeros, reducing "1.200000" to "1.2" or "3.4e1"...

23.
```
void transform_point(double & x, double & y) ;
```

Scale & shift both X & Y to graph Cartesian coordinates.

24.
```
void transform_x(double & x) ;
```

Scale and shift X value only.

25.
```
void transform_y(double & y) ;
```

Scale and shift Y value only.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Class svg_2d_plot_series

boost::svg::svg_2d_plot_series — Holds a series of 2D data values (points) to be plotted.

Synopsis

```
// In header: <boost/svg_plot/svg_2d_plot.hpp>


class svg_2d_plot_series {
public:
  // construct/copy/destruct
  template<class T > svg_2d_plot_series(T, T, std::string = "");

  // public member functions
  svg_color & area_fill() ;
  svg_2d_plot_series & area_fill(const svg_color &) ;
  svg_color & bar_area_fill() ;
  svg_2d_plot_series & bar_area_fill(const svg_color &) ;
  svg_color & bar_color() ;
  svg_2d_plot_series & bar_color(const svg_color &) ;
  bar_option bar_opt() ;
  svg_2d_plot_series & bar_opt(bar_option) ;
  double bar_width() ;
  svg_2d_plot_series & bar_width(double) ;
  bool bezier_on() ;
  svg_2d_plot_series & bezier_on(bool) ;
  svg_2d_plot_series & fill_color(const svg_color &) ;
  svg_2d_plot_series & histogram(histogram_option) ;
  int limits_count() ;
  svg_color & line_color() ;
  svg_2d_plot_series & line_color(const svg_color &) ;
  bool line_on() ;
  svg_2d_plot_series & line_on(bool) ;
  plot_line_style line_style() ;
  double line_width() ;
  svg_2d_plot_series & line_width(double) ;
  point_shape shape() ;
  svg_2d_plot_series & shape(point_shape) ;
  int size() ;
  svg_2d_plot_series & size(int) ;
  svg_2d_plot_series & stroke_color(const svg_color &) ;
  int values_count() ;
  friend svg_2d_plot_series;
  bar_style bar_style_;  // Style of bar used in histograms.
  histogram_style histogram_style_;  // Style of histogram.
  plot_point_style limit_point_style_;  // At limit data point marker. Default is cone pointing ↵
down.
  plot_line_style line_style_;  // Style (color, width...) of line joining data points.
  plot_point_style point_style_;  // Data point marker like circle, square...
  std::multimap< unc, unc > series_;  // Normal 'OK to plot' data values.
  std::multimap< double, double > series_limits_;  // 'limit' values: too big or small, or NaN.
  std::string title_;  // Title of data series (to show on legend using legend style).
};
```

Description

Data values are sorted into normal and 'at limits': NaN, infinity or too small or too large. multimap is used rather than vector of pairs because multimap sorts and ensures that lines joining data points are unaffected by the order in which data is presented. (For 1-D a vector of doubles can be used).

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

`svg_2d_plot_series` public construct/copy/destruct

1. ```
   template<class T > svg_2d_plot_series(T begin, T end, std::string title = "");
   ```

   Constructor for a data series to plot.

`svg_2d_plot_series` public member functions

1. ```
   svg_color & area_fill() ;
   ```

   Returns: area_fill info below line(s) joining data points.

2. ```
   svg_2d_plot_series & area_fill(const svg_color & col_) ;
   ```

   Set Data series area fill color.

3. ```
   svg_color & bar_area_fill() ;
   ```

   Returns: bar area fill.

4. ```
   svg_2d_plot_series & bar_area_fill(const svg_color & col) ;
   ```

   Parameters: `col` Set bar area fill color.
   Returns: reference to svg_2d_plot_series to make chainable.

5. ```
   svg_color & bar_color() ;
   ```

   Returns: bar color.

6. ```
   svg_2d_plot_series & bar_color(const svg_color & col) ;
   ```

   Set bar color.

   Returns: reference to svg_2d_plot_series to make chainable.

7. ```
   bar_option bar_opt() ;
   ```

   Returns: bar options.

8. ```
   svg_2d_plot_series & bar_opt(bar_option) ;
   ```

   Set bar option.

   Returns: reference to svg_2d_plot_series to make chainable.

9. ```
   double bar_width() ;
   ```

   Returns: bar width

10. ```
    svg_2d_plot_series & bar_width(double wid_) ;
    ```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/
    ```

Set Bar width.

Returns: reference to svg_2d_plot_series to make chainable.

11.
```
bool bezier_on() ;
```

Returns: true if line joing data points should be a bezier curve.

12.
```
svg_2d_plot_series & bezier_on(bool on_) ;
```

Set to draw bezier curved line linking data points.

Returns: reference to svg_2d_plot_series to make chainable.

13.
```
svg_2d_plot_series & fill_color(const svg_color & col_) ;
```

Set data series point marker fill color.

14.
```
svg_2d_plot_series & histogram(histogram_option opt_) ;
```

Parameters: `opt_` no_histogram = 0,

bar = +1 // Stick or column line (stroke width) vertical to X-axis.

Returns: reference to svg_2d_plot_series to make chainable.

15.
```
int limits_count() ;
```

Returns: number of values 'at limit' in a data series.

16.
```
svg_color & line_color() ;
```

Returns: color of a line to join data points.

17.
```
svg_2d_plot_series & line_color(const svg_color & col_) ;
```

Set Data series line color.

18.
```
bool line_on() ;
```

Returns: true if a line is to join data points.

19.
```
svg_2d_plot_series & line_on(bool on_) ;
```

Set to draw line linking data points.

Returns: reference to svg_2d_plot_series to make chainable.

20.
```
plot_line_style line_style() ;
```

Returns: line_style for line joining data points..

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

21.
```
double line_width() ;
```

Returns: width of line joining data points.

22.
```
svg_2d_plot_series & line_width(double wid_) ;
```

Set data series line width.

Returns: reference to svg_2d_plot_series to make chainable.

23.
```
point_shape shape() ;
```

Returns: shape of data point marker(s).

24.
```
svg_2d_plot_series & shape(point_shape shape_) ;
```

Set Data series point marker shape.

25.
```
int size() ;
```

Returns: size of data point marker(s).

26.
```
svg_2d_plot_series & size(int size_) ;
```

Set Data series point marker size.

27.
```
svg_2d_plot_series & stroke_color(const svg_color & col_) ;
```

Set Data series point marker stroke color.

28.
```
int values_count() ;
```

Returns: number of normal values in a data series.

Header <boost/svg_plot/svg_boxplot.hpp>

Create box plots in Scalable Vector Graphic (SVG) format.

Provides svg_boxplot data and function to create plots, and svg_boxplot_series to allow data values to be added. Very many functions allow fine control of the appearance and layout of plots and data markers. Items common to 1D, 2D and boxplot use axis_plot_frame. A convenient way of graphically depicting groups of numerical data through their five-number summaries. Show 1st quartile, median and 3rd quartile as a box. http://en.wikipedia.org/wiki/Boxplot

Some Implementations of the Boxplot Michael Frigge, David C. Hoaglin and Boris Iglewicz The American Statistician, Vol. 43, No. 1 (Feb., 1989), pp. 50-54

See also : The Bagplot: A Bivariate Boxplot Peter J. Rousseeuw, Ida Ruts and John W. Tukey The American Statistician, Vol. 53, No. 4 (Nov., 1999), pp. 382-387

Jacob Voytko & Paul A. Bristow

Mar 2009

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
namespace boost {
  namespace svg {
    class svg_boxplot;
    class svg_boxplot_series;
  }
}
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Class svg_boxplot

boost::svg::svg_boxplot — A plot that can display boxplots of several data series.

Synopsis

```
// In header: <boost/svg_plot/svg_boxplot.hpp>


class svg_boxplot {
public:
  // construct/copy/destruct
  svg_boxplot();

  // public member functions
  svg_boxplot & autoscale(bool) ;
  bool autoscale() ;
  svg_boxplot & autoscale_check_limits(bool) ;
  bool autoscale_check_limits() ;
  svg_boxplot & autoscale_plusminus(double) ;
  double autoscale_plusminus() ;
  bool axes_on() ;
  svg_boxplot & axes_on(bool) ;
  svg_color axis_color() ;
  svg_boxplot & axis_color(const svg_color &) ;
  double axis_width() ;
  svg_boxplot & axis_width(double) ;
  svg_color background_border_color() ;
  svg_boxplot & background_border_color(const svg_color &) ;
  double background_border_width() ;
  svg_boxplot & background_border_width(double) ;
  svg_color background_color() ;
  svg_boxplot & background_color(const svg_color &) ;
  svg_boxplot & boost_license_on(bool) ;
  bool boost_license_on() ;
  svg_color box_border() ;
  svg_boxplot & box_border(const svg_color &) ;
  svg_color box_fill() ;
  svg_boxplot & box_fill(const svg_color &) ;
  double box_width() ;
  svg_boxplot & box_width(double) ;
  void calculate_plot_window() ;
  void clear_all() ;
  int coord_precision() ;
  svg_boxplot & coord_precision(int) ;
  const std::string copyright_date() ;
  svg_boxplot & copyright_date(const std::string) ;
  const std::string copyright_holder() ;
  svg_boxplot & copyright_holder(const std::string) ;
  double data_lines_width() ;
  svg_boxplot & data_lines_width(double) ;
  const std::string & description() ;
  svg_boxplot & description(const std::string) ;
  std::string document_title() ;
  svg_boxplot & document_title(const std::string) ;
  void draw_box(double, double, double, double, const svg_style &) ;
  void draw_boxplot(svg_boxplot_series &, double) ;
  svg_boxplot &
  draw_line(double, double, double, double, const svg_color & = black) ;
  void draw_median(double, double, double, const svg_style &,
                   const value_style &) ;
  svg_boxplot &
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
draw_note(double, double, std::string, rotate_style = horizontal,
          align_style = center_align, const svg_color & = black,
          text_style & = no_style) ;
void draw_outliers(double, const std::vector< double > &,
                   const std::vector< double > &, const plot_point_style &,
                   const plot_point_style &, const value_style &) ;
svg_boxplot &
draw_plot_curve(double, double, double, double, double, double,
                const svg_color & = black) ;
svg_boxplot &
draw_plot_line(double, double, double, double, const svg_color & = black) ;
void draw_title() ;
void draw_whiskers(double, double, double, double, const svg_style &,
                   const svg_style &, const svg_style &) ;
void draw_x_axis() ;
void draw_x_axis_label() ;
void draw_x_major_tick(double, path_element &, text_element &) ;
void draw_y_axis() ;
void draw_y_axis_label() ;
void draw_y_major_tick(double, path_element &, path_element &) ;
void draw_y_minor_tick(double, path_element &, path_element &) ;
svg_color extreme_outlier_color() ;
svg_boxplot & extreme_outlier_color(const svg_color &) ;
svg_color extreme_outlier_fill() ;
svg_boxplot & extreme_outlier_fill(const svg_color &) ;
point_shape extreme_outlier_shape() ;
svg_boxplot & extreme_outlier_shape(point_shape) ;
int extreme_outlier_size() ;
svg_boxplot & extreme_outlier_size(int) ;
bool extreme_outlier_values_on() ;
svg_boxplot & extreme_outlier_values_on(bool) ;
double image_border_margin() ;
svg_boxplot & image_border_margin(double) ;
double image_border_width() ;
svg_boxplot & image_border_width(double) ;
std::pair< double, double > image_size() ;
svg_boxplot & image_x_size(unsigned int) ;
unsigned int image_x_size() ;
svg_boxplot & image_y_size(unsigned int) ;
unsigned int image_y_size() ;
svg_color legend_background_color() ;
svg_boxplot & legend_background_color(const svg_color &) ;
svg_color legend_border_color() ;
svg_boxplot & legend_border_color(const svg_color &) ;
const std::pair< double, double > legend_bottom_right() ;
bool legend_box_fill_on() ;
svg_color legend_color() ;
svg_boxplot & legend_color(const svg_color &) ;
const std::string & legend_font_family() ;
svg_boxplot & legend_font_family(const std::string &) ;
const std::string & legend_font_weight() ;
svg_boxplot & legend_font_weight(const std::string &) ;
int legend_header_font_size() ;
svg_boxplot & legend_header_font_size(int) ;
bool legend_lines() ;
svg_boxplot & legend_lines(bool) ;
bool legend_on() ;
svg_boxplot & legend_on(bool) ;
bool legend_outside() ;
legend_places legend_place() ;
svg_boxplot & legend_place(legend_places) ;
const std::string legend_title() ;
svg_boxplot & legend_title(const std::string) ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
unsigned int legend_title_font_size() ;
svg_boxplot & legend_title_font_size(unsigned int) ;
const std::pair< double, double > legend_top_left() ;
svg_boxplot & legend_top_left(double, double) ;
double legend_width() ;
svg_boxplot & legend_width(double) ;
svg_boxplot &
license(std::string = "permits", std::string = "permits",
        std::string = "requires", std::string = "permits",
        std::string = "permits") ;
const std::string license_attribution() ;
const std::string license_commercialuse() ;
const std::string license_distribution() ;
svg_boxplot & license_on(bool) ;
bool license_on() ;
const std::string license_reproduction() ;
svg_color limit_color() ;
svg_boxplot & limit_color(const svg_color &) ;
svg_color limit_fill_color() ;
svg_boxplot & limit_fill_color(const svg_color &) ;
svg_color median_color() ;
svg_boxplot & median_color(const svg_color &) ;
bool median_values_on() ;
svg_boxplot & median_values_on(bool) ;
double median_width() ;
svg_boxplot & median_width(double) ;
svg_color one_sd_color() ;
svg_boxplot & one_sd_color(const svg_color &) ;
svg_color outlier_color() ;
svg_boxplot & outlier_color(const svg_color &) ;
svg_color outlier_fill() ;
svg_boxplot & outlier_fill(const svg_color &) ;
point_shape outlier_shape() ;
svg_boxplot & outlier_shape(point_shape) ;
int outlier_size() ;
svg_boxplot & outlier_size(int) ;
svg_boxplot & outlier_style(plot_point_style &) ;
plot_point_style & outlier_style() ;
bool outlier_values_on() ;
svg_boxplot & outlier_values_on(bool) ;
template<class T >
  svg_boxplot_series & plot(const T &, const std::string & = "") ;
template<class T >
  svg_boxplot_series &
  plot(const T &, const std::string &, double, svg_style, svg_style,
       svg_style, double, svg_style, svg_style, plot_point_style,
       plot_point_style, int, value_style, text_style) ;
svg_color plot_background_color() ;
svg_boxplot & plot_background_color(const svg_color &) ;
svg_color plot_border_color() ;
svg_boxplot & plot_border_color(const svg_color &) ;
svg_boxplot & plot_border_width(double) ;
double plot_border_width() ;
bool plot_window_on() ;
svg_boxplot & plot_window_on(bool) ;
std::pair< double, double > plot_window_x() ;
svg_boxplot & plot_window_x(double, double) ;
double plot_window_x_left() ;
double plot_window_x_right() ;
std::pair< double, double > plot_window_y() ;
svg_boxplot & plot_window_y(double, double) ;
double plot_window_y_bottom() ;
double plot_window_y_top() ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
int quartile_definition() ;
svg_boxplot & quartile_definition(int) ;
svg_boxplot & size(unsigned int, unsigned int) ;
svg_color three_sd_color() ;
svg_boxplot & three_sd_color(const svg_color &) ;
svg_boxplot & title(const std::string) ;
std::string title() ;
svg_boxplot & title(const std::string &) ;
svg_color title_color() ;
svg_boxplot & title_color(const svg_color &) ;
align_style title_font_alignment() ;
svg_boxplot & title_font_alignment(align_style) ;
const std::string & title_font_decoration() ;
svg_boxplot & title_font_decoration(const std::string &) ;
const std::string & title_font_family() ;
svg_boxplot & title_font_family(const std::string &) ;
int title_font_rotation() ;
svg_boxplot & title_font_rotation(rotate_style) ;
unsigned int title_font_size() ;
svg_boxplot & title_font_size(unsigned int) ;
const std::string & title_font_stretch() ;
svg_boxplot & title_font_stretch(const std::string &) ;
const std::string & title_font_style() ;
svg_boxplot & title_font_style(const std::string &) ;
const std::string & title_font_weight() ;
svg_boxplot & title_font_weight(const std::string &) ;
bool title_on() ;
svg_boxplot & title_on(bool) ;
svg_boxplot & title_size(unsigned int) ;
void transform_x(double &) ;
void transform_y(double &) ;
svg_color two_sd_color() ;
svg_boxplot & two_sd_color(const svg_color &) ;
void update_image() ;
double whisker_length() ;
svg_boxplot & whisker_length(double) ;
svg_boxplot & write(std::ostream &) ;
svg_boxplot & write(const std::string &) ;
double x_auto_max_value() ;
double x_auto_min_value() ;
double x_auto_tick_interval() ;
int x_auto_ticks() ;
svg_boxplot & x_autoscale(const T &, const T &) ;
svg_boxplot & x_autoscale(const T &) ;
svg_boxplot & x_autoscale(std::pair< double, double >) ;
svg_boxplot & x_autoscale(bool) ;
bool x_autoscale() ;
svg_color x_axis_color() ;
svg_boxplot & x_axis_color(const svg_color &) ;
svg_color x_axis_label_color() ;
svg_boxplot & x_axis_label_color(const svg_color &) ;
bool x_axis_on() ;
svg_boxplot & x_axis_on(bool) ;
double x_axis_position() ;
svg_boxplot & x_axis_position(int) ;
double x_axis_width() ;
svg_boxplot & x_axis_width(double) ;
svg_boxplot &
x_decor(const std::string &, const std::string & = "",
        const std::string & = "") ;
svg_color x_df_color() ;
svg_boxplot & x_df_color(const svg_color &) ;
bool x_df_on() ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
svg_boxplot & x_df_on(bool) ;
std::string x_label() ;
svg_boxplot & x_label(const std::string &) ;
svg_color x_label_color() ;
svg_boxplot & x_label_color(const svg_color &) ;
const std::string & x_label_font_family() ;
svg_boxplot & x_label_font_family(const std::string &) ;
unsigned int x_label_font_size() ;
svg_boxplot & x_label_font_size(unsigned int) ;
bool x_label_on() ;
svg_boxplot & x_label_on(bool) ;
svg_boxplot & x_label_size(unsigned int) ;
std::string x_label_text() ;
std::string x_label_units() ;
svg_boxplot & x_label_units(const std::string &) ;
bool x_label_units_on() ;
svg_boxplot & x_label_units_on(bool) ;
double x_label_width() ;
svg_boxplot & x_label_width(double) ;
svg_boxplot & x_labels_strip_e0s(bool) ;
svg_color x_major_grid_color() ;
svg_boxplot & x_major_grid_color(const svg_color &) ;
bool x_major_grid_on() ;
svg_boxplot & x_major_grid_on(bool) ;
double x_major_grid_width() ;
svg_boxplot & x_major_grid_width(double) ;
double x_major_interval() ;
svg_boxplot & x_major_interval(double) ;
rotate_style x_major_label_rotation() ;
svg_boxplot & x_major_label_rotation(rotate_style) ;
int x_major_labels() ;
svg_boxplot & x_major_labels_on(int) ;
int x_major_labels_side() ;
svg_boxplot & x_major_labels_side(int) ;
double x_major_tick() ;
svg_boxplot & x_major_tick(double) ;
svg_color x_major_tick_color() ;
svg_boxplot & x_major_tick_color(const svg_color &) ;
double x_major_tick_length() ;
svg_boxplot & x_major_tick_length(double) ;
svg_boxplot & x_major_tick_width(double) ;
double x_major_tick_width() ;
double x_max() ;
svg_boxplot & x_max(double) ;
double x_min() ;
svg_boxplot & x_min(double) ;
int x_min_ticks() ;
svg_boxplot & x_min_ticks(int) ;
svg_color x_minor_grid_color() ;
svg_boxplot & x_minor_grid_color(const svg_color &) ;
bool x_minor_grid_on() ;
svg_boxplot & x_minor_grid_on(bool) ;
double x_minor_grid_width() ;
svg_boxplot & x_minor_grid_width(double) ;
svg_boxplot & x_minor_interval(double) ;
double x_minor_interval() ;
svg_color x_minor_tick_color() ;
svg_boxplot & x_minor_tick_color(const svg_color &) ;
double x_minor_tick_length() ;
svg_boxplot & x_minor_tick_length(double) ;
double x_minor_tick_width() ;
svg_boxplot & x_minor_tick_width(double) ;
unsigned int x_num_minor_ticks() ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
svg_boxplot & x_num_minor_ticks(unsigned int) ;
svg_color x_plusminus_color() ;
svg_boxplot & x_plusminus_color(const svg_color &) ;
bool x_plusminus_on() ;
svg_boxplot & x_plusminus_on(bool) ;
const std::string x_prefix() ;
std::pair< double, double > x_range() ;
svg_boxplot & x_range(double, double) ;
const std::string x_separator() ;
int x_steps() ;
svg_boxplot & x_steps(int) ;
const std::string x_suffix() ;
svg_color x_tick_color() ;
svg_boxplot & x_tick_color(const svg_color &) ;
double x_tick_length() ;
svg_boxplot & x_tick_length(unsigned int) ;
svg_boxplot & x_tick_width(unsigned int) ;
bool x_ticks_down_on() ;
svg_boxplot & x_ticks_down_on(bool) ;
int x_ticks_on_window_or_axis() ;
svg_boxplot & x_ticks_on_window_or_axis(int) ;
bool x_ticks_up_on() ;
svg_boxplot & x_ticks_up_on(bool) ;
svg_color x_ticks_values_color() ;
svg_boxplot & x_ticks_values_color(const svg_color &) ;
const std::string & x_ticks_values_font_family() ;
svg_boxplot & x_ticks_values_font_family(const std::string &) ;
unsigned int x_ticks_values_font_size() ;
svg_boxplot & x_ticks_values_font_size(unsigned int) ;
std::ios_base::fmtflags x_ticks_values_ioflags() ;
svg_boxplot & x_ticks_values_ioflags(std::ios_base::fmtflags) ;
int x_ticks_values_precision() ;
svg_boxplot & x_ticks_values_precision(int) ;
double x_tight() ;
svg_boxplot & x_tight(double) ;
unsigned int x_value_font_size() ;
svg_boxplot & x_value_font_size(unsigned int) ;
int x_value_ioflags() ;
svg_boxplot & x_value_ioflags(int) ;
int x_value_precision() ;
svg_boxplot & x_value_precision(int) ;
svg_color x_values_color() ;
svg_boxplot & x_values_color(const svg_color &) ;
const std::string & x_values_font_family() ;
svg_boxplot & x_values_font_family(const std::string &) ;
unsigned int x_values_font_size() ;
svg_boxplot & x_values_font_size(unsigned int) ;
std::ios_base::fmtflags x_values_ioflags() ;
svg_boxplot & x_values_ioflags(std::ios_base::fmtflags) ;
bool x_values_on() ;
svg_boxplot & x_values_on(bool) ;
int x_values_precision() ;
svg_boxplot & x_values_precision(int) ;
int x_values_rotation() ;
svg_boxplot & x_values_rotation(rotate_style) ;
bool x_with_zero() ;
svg_boxplot & x_with_zero(bool) ;
template<class T > svg_boxplot & y_autoscale(const T &) ;
template<class T > svg_boxplot & y_autoscale(const T &, const T &) ;
svg_boxplot & y_autoscale(std::pair< double, double >) ;
svg_boxplot & y_autoscale(double, double) ;
svg_boxplot & y_autoscale(bool) ;
bool y_autoscale() ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
  svg_color y_axis_color() ;
  svg_boxplot & y_axis_color(const svg_color &) ;
  bool y_axis_on() ;
  svg_boxplot & y_axis_on(bool) ;
  double y_axis_position() ;
  svg_boxplot & y_axis_position(int) ;
  std::string y_label() ;
  svg_boxplot & y_label(const std::string &) ;
  svg_color y_label_color() ;
  svg_boxplot & y_label_color(const svg_color &) ;
  svg_boxplot & y_label_font_size(unsigned int) ;
  svg_boxplot & y_label_on(bool) ;
  std::string y_label_text() ;
  std::string y_label_units() ;
  svg_boxplot & y_label_units(const std::string &) ;
  bool y_labels_strip_e0s() ;
  svg_boxplot & y_major_interval(double) ;
  svg_boxplot & y_major_labels_on(int) ;
  svg_boxplot & y_major_tick_color(const svg_color &) ;
  svg_boxplot & y_major_tick_length(unsigned int) ;
  svg_boxplot & y_major_tick_width(unsigned int) ;
  double y_minor_interval() ;
  svg_boxplot & y_minor_tick_color(const svg_color &) ;
  svg_boxplot & y_minor_tick_length(unsigned int) ;
  svg_boxplot & y_minor_tick_width(unsigned int) ;
  svg_boxplot & y_num_minor_ticks(unsigned int) ;
  std::pair< double, double > y_range() ;
  svg_boxplot & y_range(double, double) ;

  // protected member functions
  void adjust_limits(double &, double &) ;
  void clear_background() ;
  void clear_grids() ;
  void clear_legend() ;
  void clear_plot_background() ;
  void clear_points() ;
  void clear_title() ;
  void clear_x_axis() ;
  void clear_y_axis() ;
  void draw_legend() ;
  void draw_plot_point(double, double, g_element &, plot_point_style &, unc,
                       unc) ;
  void draw_plot_point_value(double, double, g_element &, value_style &,
                             plot_point_style &, unc) ;
  void draw_plot_point_values(double, double, g_element &, g_element &,
                              const value_style &, const value_style &, unc,
                              unc) ;
  void draw_x_major_tick(double, path_element &, path_element &) ;
  void draw_x_minor_tick(double, path_element &, path_element &) ;
  void place_legend_box() ;
  void size_legend_box() ;
  std::string sv(double, const value_style &, bool = false) ;
  void transform_point(double &, double &) ;
  text_style a_style_;
  bool autoscale_check_limits_;
  double autoscale_plusminus_;  // For uncertain values, allow for plusminus ellipses showing ↵
67%, 95% and 99% confidence limits.
 < For example, if a max value is 1.2 +or- 0.02, then 1.4 will be used for autoscaling the max↵
imum.
 < Similarly, if a min value is 1.2 +or- 0.02, then 1.0 will be used for autoscaling the minimum.
  svg_style axis_style_;
  svg_style box_style_;
  double box_width_;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
plot_point_style ext_outlier_;
bool extreme_outlier_values_on_;
svg image_;
box_style image_border_;
svg_style max_whisker_style_;
svg_style median_style_;
bool median_values_on_;
plot_point_style mild_outlier_;
svg_style min_whisker_style_;
bool outlier_values_on_;
double plot_bottom_;
double plot_left_;  // < Intersection of Y axis, or not.
double plot_right_;
double plot_top_;
box_style plot_window_border_;
std::string plot_window_clip_;
bool plot_window_on_;
text_style point_symbols_style_;
int quartile_definition_;
std::vector< svg_boxplot_series > series;
text_style series_style_;
friend svg_boxplot_series;
double text_margin_;
text_element title_info_;
bool title_on_;
text_style title_style_;
text_style value_style_;
value_style values_style_;
double whisker_length_;
double x_auto_max_value_;
double x_auto_min_value_;
double x_auto_tick_interval_;
int x_auto_ticks_;
bool x_autoscale_;
axis_line_style x_axis_;
text_style x_axis_label_style_;
double x_axis_position_;  // and y coordinates increase DOWN the page!
bool x_include_zero_;
text_element x_label_info_;
int x_min_ticks_;
double x_scale_;
double x_shift_;  // Used by function transform().
int x_steps_;
ticks_labels_style x_ticks_;
double x_tight_;
text_element x_units_info_;
text_style x_value_label_style_;
double y_auto_max_value_;
double y_auto_min_value_;
double y_auto_tick_interval_;
int y_auto_ticks_;
bool y_autoscale_;
axis_line_style y_axis_;
text_style y_axis_label_style_;
double y_axis_position_;  // < Intersection of X axis, or not.
bool y_include_zero_;
text_element y_label_info_;
int y_min_ticks_;
double y_scale_;  // to go from Cartesian to SVG coordinates.
double y_shift_;  // SVG coordinates origin is at top left,.
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
  int y_steps_;
  ticks_labels_style y_ticks_;
  double y_tight_;
  text_element y_units_info_;
  text_style y_value_label_style_;
};
```

Description

Holds all data about the plot (but not any data series - see svg_boxplot_series) axis_plot_frame.hpp contains functions common to 1 and 2-D, and boxplot.

`svg_boxplot` public construct/copy/destruct

1. ```
   svg_boxplot();
   ```

   Default constructor providing all the default colors, style etc,

`svg_boxplot` public member functions

1. ```
   svg_boxplot & autoscale(bool b) ;
   ```

2. ```
   bool autoscale() ;
   ```

3. ```
   svg_boxplot & autoscale_check_limits(bool b) ;
   ```

4. ```
   bool autoscale_check_limits() ;
   ```

5. ```
   svg_boxplot & autoscale_plusminus(double) ;
   ```

6. ```
   double autoscale_plusminus() ;
   ```

7. ```
   bool axes_on() ;
   ```

8. ```
   svg_boxplot & axes_on(bool is) ;
   ```

9. ```
   svg_color axis_color() ;
   ```

   Returns: Color of vertical whisker axis line in box.

10. ```
    svg_boxplot & axis_color(const svg_color & color) ;
    ```

    Set color of vertical whisker axis line in box.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/
    ```

Returns: reference to svg_boxplot to make chainable.

11.
```
double axis_width() ;
```

Returns: Width of the box, not the border.

12.
```
svg_boxplot & axis_width(double l) ;
```

Set width of the box, not the border.

Returns: reference to svg_boxplot to make chainable.

13.
```
svg_color background_border_color() ;
```

Returns: Color of the border of the background for the SVG image.

14.
```
svg_boxplot & background_border_color(const svg_color & col) ;
```

Set SVG image background border color.

Returns: reference to svg_boxplot to make chainable.

15.
```
double background_border_width() ;
```

16.
```
svg_boxplot & background_border_width(double w) ;
```

17.
```
svg_color background_color() ;
```

Returns: Color of the background for the SVG image.

18.
```
svg_boxplot & background_color(const svg_color & col) ;
```

Set SVG image background color.

19.
```
svg_boxplot & boost_license_on(bool l) ;
```

20.
```
bool boost_license_on() ;
```

21.
```
svg_color box_border() ;
```

Returns: Color of box border.

22.
```
svg_boxplot & box_border(const svg_color & color) ;
```

Set color of box border.

Returns: reference to svg_boxplot to make chainable.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

23.
```
svg_color box_fill() ;
```

Returns: Color of box fill (not border).

24.
```
svg_boxplot & box_fill(const svg_color & color) ;
```

Set color of box fill (not border).

Returns: reference to svg_boxplot to make chainable.

25.
```
double box_width() ;
```

Width of the actual box (not the border).

Returns: width of the box. (Width of the box, not the border).

26.
```
svg_boxplot & box_width(double width) ;
```

Set width of the box. (Width of the box, not the border).

Returns: reference to svg_boxplot to make chainable.

27.
```
void calculate_plot_window() ;
```

Calculate the position of the plot window.

28.
```
void clear_all() ;
```

Clear all the previous information and rebuild the SVG image. When writing to multiple documents, the contents of the plot may change significantly between. Rather than figuring out what has and has not changed, just erase the contents of the legend, title... in the document and start over.

29.
```
int coord_precision() ;
```

30.
```
svg_boxplot & coord_precision(int digits) ;
```

31.
```
const std::string copyright_date() ;
```

32.
```
svg_boxplot & copyright_date(const std::string d) ;
```

33.
```
const std::string copyright_holder() ;
```

34.
```
svg_boxplot & copyright_holder(const std::string d) ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

35.
```
double data_lines_width() ;
```

36.
```
svg_boxplot & data_lines_width(double width) ;
```

37.
```
const std::string & description() ;
```

38.
```
svg_boxplot & description(const std::string d) ;
```

39.
```
std::string document_title() ;
```

40.
```
svg_boxplot & document_title(const std::string d) ;
```

41.
```
void draw_box(double q1, double q3, double x, double width,
              const svg_style & box_styl) ;
```

Draw the box border and any fill.

42.
```
void draw_boxplot(svg_boxplot_series & series, double x_offset) ;
```

Draw a whole boxplot, box, median line, axis whiskers, and outliers.

43.
```
svg_boxplot &
draw_line(double x1, double y1, double x2, double y2,
          const svg_color & col = black) ;
```

44.
```
void draw_median(double median, double x_offset, double box_width,
                 const svg_style & median_style,
                 const value_style & values_style) ;
```

Draw the median of the data series line within the box, and optionally the median value.

45.
```
svg_boxplot &
draw_note(double x, double y, std::string note, rotate_style rot = horizontal,
          align_style al = center_align, const svg_color & = black,
          text_style & tsty = no_style) ;
```

46.
```
void draw_outliers(double x, const std::vector< double > & outliers,
                   const std::vector< double > & extreme_outliers,
                   const plot_point_style & mild_style,
                   const plot_point_style & extreme_style,
                   const value_style & values_style) ;
```

Draw any outliers, both mild and extreme.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

47.
```
svg_boxplot &
draw_plot_curve(double x1, double y1, double x2, double y2, double x3,
                double y3, const svg_color & col = black) ;
```

48.
```
svg_boxplot &
draw_plot_line(double x1, double y1, double x2, double y2,
               const svg_color & col = black) ;
```

49.
```
void draw_title() ;
```

Update title_info_ with position.

Assumes align = center_align. Note: center_align will ensure that will center correctly even if original string is long because contains Unicode like because the browser render engine does the centering.

50.
```
void draw_whiskers(double min, double max, double length, double x,
                   const svg_style & min_whisker,
                   const svg_style & max_whisker,
                   const svg_style & axis_whisker) ;
```

Draw the whiskers.

51.
```
void draw_x_axis() ;
```

Draw the horizontal X-axis line.

52.
```
void draw_x_axis_label() ;
```

Draw an axis label (and optional units) for example "length (km)".

53.
```
void draw_x_major_tick(double i, path_element & tick_path,
                       text_element & series_info) ;
```

Draw X axis major tick, and optional boxplot label.

54.
```
void draw_y_axis() ;
```

Draw the vertical Y-axis line.

55.
```
void draw_y_axis_label() ;
```

Draw vertical y_axis label, and optional y units.

56.
```
void draw_y_major_tick(double value, path_element & tick_path,
                       path_element & grid_path) ;
```

Draw a Y axis major tick, tick value labels & horizontal grid.

57.
```
void draw_y_minor_tick(double value, path_element & tick_path,
                       path_element & grid_path) ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Draw a Y-axis minor tick and optional grid.

58.
```
svg_color extreme_outlier_color() ;
```

Returns: Color of extreme outlier.

59.
```
svg_boxplot & extreme_outlier_color(const svg_color & color) ;
```

Color of extreme outlier.

Returns: Fill color of outlier marker.
Returns: reference to svg_boxplot to make chainable.

60.
```
svg_color extreme_outlier_fill() ;
```

Returns: Color of extreme outlier fill.

61.
```
svg_boxplot & extreme_outlier_fill(const svg_color & color) ;
```

Set color of extreme outlier fill.

Returns: Color of extreme outlier marker.
Returns: reference to svg_boxplot to make chainable.

62.
```
point_shape extreme_outlier_shape() ;
```

Shape of extreme outlier marker.

Returns: Shape of extreme outlier marker.

63.
```
svg_boxplot & extreme_outlier_shape(point_shape shape) ;
```

Set shape of extreme outlier marker.

Returns: reference to svg_boxplot to make chainable.

64.
```
int extreme_outlier_size() ;
```

Returns: size of extreme outlier marker.

65.
```
svg_boxplot & extreme_outlier_size(int size) ;
```

Set size of extreme outlier marker.

Returns: Shape of extreme outlier marker.
Returns: reference to svg_boxplot to make chainable.

66.
```
bool extreme_outlier_values_on() ;
```

Returns: true if to show extreme outlier value(s).

67.
```
svg_boxplot & extreme_outlier_values_on(bool cmd) ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Set true to show extreme outlier values.

Returns: reference to svg_boxplot to make chainable.

68.
```
double image_border_margin() ;
```

69.
```
svg_boxplot & image_border_margin(double w) ;
```

70.
```
double image_border_width() ;
```

71.
```
svg_boxplot & image_border_width(double w) ;
```

72.
```
std::pair< double, double > image_size() ;
```

73.
```
svg_boxplot & image_x_size(unsigned int i) ;
```

74.
```
unsigned int image_x_size() ;
```

Returns: width of the SVG image.

75.
```
svg_boxplot & image_y_size(unsigned int i) ;
```

76.
```
unsigned int image_y_size() ;
```

Returns: height of the SVG image.

77.
```
svg_color legend_background_color() ;
```

78.
```
svg_boxplot & legend_background_color(const svg_color & col) ;
```

79.
```
svg_color legend_border_color() ;
```

80.
```
svg_boxplot & legend_border_color(const svg_color & col) ;
```

81.
```
const std::pair< double, double > legend_bottom_right() ;
```

82.
```
bool legend_box_fill_on() ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

83.
```
svg_color legend_color() ;
```

84.
```
svg_boxplot & legend_color(const svg_color & col) ;
```

85.
```
const std::string & legend_font_family() ;
```

86.
```
svg_boxplot & legend_font_family(const std::string & family) ;
```

87.
```
const std::string & legend_font_weight() ;
```

88.
```
svg_boxplot & legend_font_weight(const std::string & weight) ;
```

89.
```
int legend_header_font_size() ;
```

90.
```
svg_boxplot & legend_header_font_size(int size) ;
```

91.
```
bool legend_lines() ;
```

92.
```
svg_boxplot & legend_lines(bool is) ;
```

93.
```
bool legend_on() ;
```

94.
```
svg_boxplot & legend_on(bool cmd) ;
```

95.
```
bool legend_outside() ;
```

96.
```
legend_places legend_place() ;
```

97.
```
svg_boxplot & legend_place(legend_places l) ;
```

98.
```
const std::string legend_title() ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

99.
```
svg_boxplot & legend_title(const std::string title) ;
```

100
```
unsigned int legend_title_font_size() ;
```

101.
```
svg_boxplot & legend_title_font_size(unsigned int size) ;
```

102
```
const std::pair< double, double > legend_top_left() ;
```

103
```
svg_boxplot & legend_top_left(double x, double y) ;
```

104
```
double legend_width() ;
```

105
```
svg_boxplot & legend_width(double width) ;
```

106
```
svg_boxplot &
license(std::string repro = "permits", std::string distrib = "permits",
        std::string attrib = "requires", std::string commercial = "permits",
        std::string derivative = "permits") ;
```

107.
```
const std::string license_attribution() ;
```

108
```
const std::string license_commercialuse() ;
```

109
```
const std::string license_distribution() ;
```

110
```
svg_boxplot & license_on(bool l) ;
```

111.
```
bool license_on() ;
```

112
```
const std::string license_reproduction() ;
```

113
```
svg_color limit_color() ;
```

114
```
svg_boxplot & limit_color(const svg_color &) ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

115
```
svg_color limit_fill_color() ;
```

116
```
svg_boxplot & limit_fill_color(const svg_color &) ;
```

117.
```
svg_color median_color() ;
```

Returns: color of median line in box.

118
```
svg_boxplot & median_color(const svg_color & color) ;
```

Set color of median line in box.

Returns: reference to svg_boxplot to make chainable.

119
```
bool median_values_on() ;
```

Returns: true if to show median value(s).

120
```
svg_boxplot & median_values_on(bool cmd) ;
```

Set true to show median value(s).

Returns: reference to svg_boxplot to make chainable.

121.
```
double median_width() ;
```

Returns: width of the box (not the border).

122
```
svg_boxplot & median_width(double l) ;
```

Set width of the box (not the border).

Returns: reference to svg_boxplot to make chainable.

123
```
svg_color one_sd_color() ;
```

124
```
svg_boxplot & one_sd_color(const svg_color &) ;
```

125
```
svg_color outlier_color() ;
```

Set Color of outlier marker.

Returns: Color of mild outlier.

126
```
svg_boxplot & outlier_color(const svg_color & color) ;
```

Set color of outlier.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Returns: reference to svg_boxplot to make chainable.

127.
```
svg_color outlier_fill() ;
```

Set Fill color of outlier marker.

Returns: Color of outlier fill.

128
```
svg_boxplot & outlier_fill(const svg_color & color) ;
```

Set color of mild outlier fill.

Returns: Color of outlier marker.
Returns: reference to svg_boxplot to make chainable.

129
```
point_shape outlier_shape() ;
```

Returns: Outlier marker shape.

130
```
svg_boxplot & outlier_shape(point_shape shape) ;
```

Set shape of outlier marker.

Returns: Fill color of extreme outlier marker.
Returns: reference to svg_boxplot to make chainable.

131
```
int outlier_size() ;
```

Set size of outlier marker.

Returns: Size of outlier marker.

132
```
svg_boxplot & outlier_size(int size) ;
```

Set size of outlier marker.

Returns: Shape of outlier marker.
Returns: reference to svg_boxplot to make chainable.

133
```
svg_boxplot & outlier_style(plot_point_style & os) ;
```

134
```
plot_point_style & outlier_style() ;
```

135
```
bool outlier_values_on() ;
```

Returns: true if to show mild outlier value(s).

136
```
svg_boxplot & outlier_values_on(bool cmd) ;
```

Set true to show mild outlier values.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Returns: reference to svg_boxplot to make chainable.

137.
```
template<class T >
  svg_boxplot_series &
  plot(const T & container, const std::string & title = "") ;
```

plot version copying box'n'whiskers parameters from parent boxplot.

138
```
template<class T >
  svg_boxplot_series &
  plot(const T & container, const std::string & title, double bw,
       svg_style bs, svg_style ms, svg_style as, double wl, svg_style minws,
       svg_style maxws, plot_point_style os, plot_point_style extos,
       int q_def, value_style vs, text_style ss) ;
```

< Add a data series (the whole container) to boxplot.

139
```
svg_color plot_background_color() ;
```

Returns: Color of the background for the plot.

140
```
svg_boxplot & plot_background_color(const svg_color & col) ;
```

Set plot window background color.

Returns: reference to svg_boxplot to make chainable.

141.
```
svg_color plot_border_color() ;
```

Returns: Color of the border of the background for the plot.

142
```
svg_boxplot & plot_border_color(const svg_color & col) ;
```

Set plot window border color.

Returns: reference to svg_boxplot to make chainable.

143
```
svg_boxplot & plot_border_width(double w) ;
```

144
```
double plot_border_width() ;
```

145
```
bool plot_window_on() ;
```

146
```
svg_boxplot & plot_window_on(bool cmd) ;
```

147.
```
std::pair< double, double > plot_window_x() ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
svg_boxplot & plot_window_x(double min_x, double max_x) ;
```

```
double plot_window_x_left() ;
```

```
double plot_window_x_right() ;
```

```
std::pair< double, double > plot_window_y() ;
```

```
svg_boxplot & plot_window_y(double min_y, double max_y) ;
```

```
double plot_window_y_bottom() ;
```

```
double plot_window_y_top() ;
```

```
int quartile_definition() ;
```

Returns: definition # of quartile.

```
svg_boxplot & quartile_definition(int def) ;
```

Set definition of quartile.

Returns: Size of extreme outlier marker.
Returns: reference to svg_boxplot to make chainable.

```
svg_boxplot & size(unsigned int x, unsigned int y) ;
```

Set SVG image width (x) and height (y).

Returns: reference to svg_boxplot to make chainable.

```
svg_color three_sd_color() ;
```

```
svg_boxplot & three_sd_color(const svg_color &) ;
```

```
svg_boxplot & title(const std::string title) ;
```

```
std::string title() ;
```

Returns: title of the plot.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
svg_boxplot & title(const std::string & str) ;
```

Set title text for plot.

Returns: reference to svg_boxplot to make chainable.

```
svg_color title_color() ;
```

Returns: color of the title.

```
svg_boxplot & title_color(const svg_color & col) ;
```

Set boxplot title color.

Returns: reference to svg_boxplot to make chainable.

```
align_style title_font_alignment() ;
```

```
svg_boxplot & title_font_alignment(align_style alignment) ;
```

```
const std::string & title_font_decoration() ;
```

```
svg_boxplot & title_font_decoration(const std::string & decoration) ;
```

```
const std::string & title_font_family() ;
```

```
svg_boxplot & title_font_family(const std::string & family) ;
```

```
int title_font_rotation() ;
```

```
svg_boxplot & title_font_rotation(rotate_style rotate) ;
```

```
unsigned int title_font_size() ;
```

```
svg_boxplot & title_font_size(unsigned int i) ;
```

```
const std::string & title_font_stretch() ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
svg_boxplot & title_font_stretch(const std::string & stretch) ;
```

```
const std::string & title_font_style() ;
```

```
svg_boxplot & title_font_style(const std::string & style) ;
```

```
const std::string & title_font_weight() ;
```

```
svg_boxplot & title_font_weight(const std::string & weight) ;
```

```
bool title_on() ;
```

```
svg_boxplot & title_on(bool cmd) ;
```

Write SVG boxplot to ostream.

Set true to show whole boxplot title.

Returns: reference to svg_boxplot to make chainable.

```
svg_boxplot & title_size(unsigned int size) ;
```

Set font size for title text.

Returns: reference to svg_boxplot to make chainable.

```
void transform_x(double & x) ;
```

Transform X coordinate from Cartesian to SVG.

```
void transform_y(double & y) ;
```

Transform Y coordinate from Cartesian to SVG.

```
svg_color two_sd_color() ;
```

```
svg_boxplot & two_sd_color(const svg_color &) ;
```

```
void update_image() ;
```

Update the entire SVG image.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
double whisker_length() ;
```

```
svg_boxplot & whisker_length(double width) ;
```

Set the length of the whisker.

Returns: reference to svg_boxplot to make chainable.

```
svg_boxplot & write(std::ostream & s_out) ;
```

Write SVG boxplot to file.

Write SVG image to ostream.

Returns: reference to svg_boxplot to make chainable.

```
svg_boxplot & write(const std::string & file) ;
```

Write SVG image to file.

```
double x_auto_max_value() ;
```

```
double x_auto_min_value() ;
```

```
double x_auto_tick_interval() ;
```

```
int x_auto_ticks() ;
```

```
svg_boxplot & x_autoscale(const T & begin, const T & end) ;
```

autoscale X axis using the whole data series.

```
svg_boxplot & x_autoscale(const T & container) ;
```

autoscale X axis using a pair of doubles.

```
svg_boxplot & x_autoscale(std::pair< double, double > p) ;
```

```
svg_boxplot & x_autoscale(bool b) ;
```

```
bool x_autoscale() ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
svg_color x_axis_color() ;
```

```
svg_boxplot & x_axis_color(const svg_color & col) ;
```

```
svg_color x_axis_label_color() ;
```

```
svg_boxplot & x_axis_label_color(const svg_color & col) ;
```

```
bool x_axis_on() ;
```

```
svg_boxplot & x_axis_on(bool is) ;
```

```
double x_axis_position() ;
```

Returns: position of the horizontal X-axis line (on the border).

```
svg_boxplot & x_axis_position(int pos) ;
```

Set position of the horizontal X-axis line (on the border). But controlled by the intersection with Y-axis, so this only changes the default position from bottom to top, but will be changed if X-axis intersects the Y-axis (that is if Y-axis includes zero).

Returns: Width of the box (not the border).
Returns: reference to svg_boxplot to make chainable.

```
double x_axis_width() ;
```

```
svg_boxplot & x_axis_width(double width) ;
```

```
svg_boxplot &
x_decor(const std::string & pre, const std::string & sep = "",
        const std::string & suf = "") ;
```

```
svg_color x_df_color() ;
```

```
svg_boxplot & x_df_color(const svg_color & col) ;
```

```
bool x_df_on() ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

216
```
svg_boxplot & x_df_on(bool b) ;
```

217
```
std::string x_label() ;
```

218
```
svg_boxplot & x_label(const std::string & str) ;
```

Set label for X axis (can also append optional units).

x_label_units_on and

x_label_units
Returns: reference to svg_boxplot to make chainable.

219
```
svg_color x_label_color() ;
```

Returns: Color of the X axis label.

220
```
svg_boxplot & x_label_color(const svg_color & col) ;
```

Set the font color for the X axis label.

Returns: reference to svg_boxplot to make chainable.

221
```
const std::string & x_label_font_family() ;
```

222
```
svg_boxplot & x_label_font_family(const std::string & family) ;
```

223
```
unsigned int x_label_font_size() ;
```

224
```
svg_boxplot & x_label_font_size(unsigned int i) ;
```

225
```
bool x_label_on() ;
```

Returns: label for the X axis.

226
```
svg_boxplot & x_label_on(bool cmd) ;
```

true if to include title in plot.

Set true if X axis name or label, for example: "length of thing".

Returns: reference to svg_boxplot to make chainable.

227
```
svg_boxplot & x_label_size(unsigned int size) ;
```

Set the font size for the X axis label.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Returns: reference to svg_boxplot to make chainable.

```
std::string x_label_text() ;
```

Returns: Text of label for X axis.

```
std::string x_label_units() ;
```

```
svg_boxplot & x_label_units(const std::string & str) ;
```

```
bool x_label_units_on() ;
```

```
svg_boxplot & x_label_units_on(bool cmd) ;
```

```
double x_label_width() ;
```

```
svg_boxplot & x_label_width(double width) ;
```

```
svg_boxplot & x_labels_strip_e0s(bool cmd) ;
```

```
svg_color x_major_grid_color() ;
```

```
svg_boxplot & x_major_grid_color(const svg_color & col) ;
```

```
bool x_major_grid_on() ;
```

```
svg_boxplot & x_major_grid_on(bool is) ;
```

```
double x_major_grid_width() ;
```

```
svg_boxplot & x_major_grid_width(double w) ;
```

```
double x_major_interval() ;
```

```
svg_boxplot & x_major_interval(double inter) ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
rotate_style x_major_label_rotation() ;
```

```
svg_boxplot & x_major_label_rotation(rotate_style rot) ;
```

```
int x_major_labels() ;
```

Returns: which side of the X axis for labels.

```
svg_boxplot & x_major_labels_on(int cmd) ;
```

Set direction of X major labels. < 0 means to down (default), 0 (false) means none, > 0 means to top.

Returns: reference to svg_boxplot to make chainable.

```
int x_major_labels_side() ;
```

```
svg_boxplot & x_major_labels_side(int cmd) ;
```

```
double x_major_tick() ;
```

```
svg_boxplot & x_major_tick(double d) ;
```

```
svg_color x_major_tick_color() ;
```

```
svg_boxplot & x_major_tick_color(const svg_color & col) ;
```

```
double x_major_tick_length() ;
```

```
svg_boxplot & x_major_tick_length(double length) ;
```

```
svg_boxplot & x_major_tick_width(double width) ;
```

```
double x_major_tick_width() ;
```

Returns: Width of major ticks on the X axis.

```
double x_max() ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
svg_boxplot & x_max(double x) ;
```

```
double x_min() ;
```

```
svg_boxplot & x_min(double min_x) ;
```

```
int x_min_ticks() ;
```

```
svg_boxplot & x_min_ticks(int min_ticks) ;
```

```
svg_color x_minor_grid_color() ;
```

```
svg_boxplot & x_minor_grid_color(const svg_color & col) ;
```

```
bool x_minor_grid_on() ;
```

```
svg_boxplot & x_minor_grid_on(bool is) ;
```

```
double x_minor_grid_width() ;
```

```
svg_boxplot & x_minor_grid_width(double w) ;
```

```
svg_boxplot & x_minor_interval(double interval) ;
```

```
double x_minor_interval() ;
```

```
svg_color x_minor_tick_color() ;
```

```
svg_boxplot & x_minor_tick_color(const svg_color & col) ;
```

```
double x_minor_tick_length() ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
svg_boxplot & x_minor_tick_length(double length) ;
```

```
double x_minor_tick_width() ;
```

```
svg_boxplot & x_minor_tick_width(double width) ;
```

```
unsigned int x_num_minor_ticks() ;
```

```
svg_boxplot & x_num_minor_ticks(unsigned int num) ;
```

```
svg_color x_plusminus_color() ;
```

```
svg_boxplot & x_plusminus_color(const svg_color & col) ;
```

```
bool x_plusminus_on() ;
```

```
svg_boxplot & x_plusminus_on(bool b) ;
```

```
const std::string x_prefix() ;
```

```
std::pair< double, double > x_range() ;
```

```
svg_boxplot & x_range(double min_x, double max_x) ;
```

```
const std::string x_separator() ;
```

```
int x_steps() ;
```

```
svg_boxplot & x_steps(int steps) ;
```

```
const std::string x_suffix() ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
svg_color x_tick_color() ;
```

Returns: Color of ticks on the X axis.

```
svg_boxplot & x_tick_color(const svg_color & col) ;
```

Returns: Y major ticks color.

reference to svg_boxplot to make chainable.

```
double x_tick_length() ;
```

Returns: Length of major ticks on the X axis.

```
svg_boxplot & x_tick_length(unsigned int length) ;
```

Set the length of major ticks on the X axis.

Returns: reference to svg_boxplot to make chainable.

```
svg_boxplot & x_tick_width(unsigned int width) ;
```

Set the width of major ticks on the X axis.

Returns: reference to svg_boxplot to make chainable.

```
bool x_ticks_down_on() ;
```

```
svg_boxplot & x_ticks_down_on(bool cmd) ;
```

```
int x_ticks_on_window_or_axis() ;
```

```
svg_boxplot & x_ticks_on_window_or_axis(int cmd) ;
```

```
bool x_ticks_up_on() ;
```

```
svg_boxplot & x_ticks_up_on(bool cmd) ;
```

```
svg_color x_ticks_values_color() ;
```

```
svg_boxplot & x_ticks_values_color(const svg_color & col) ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
const std::string & x_ticks_values_font_family() ;
```

```
svg_boxplot & x_ticks_values_font_family(const std::string & family) ;
```

```
unsigned int x_ticks_values_font_size() ;
```

```
svg_boxplot & x_ticks_values_font_size(unsigned int i) ;
```

```
std::ios_base::fmtflags x_ticks_values_ioflags() ;
```

```
svg_boxplot & x_ticks_values_ioflags(std::ios_base::fmtflags f) ;
```

```
int x_ticks_values_precision() ;
```

```
svg_boxplot & x_ticks_values_precision(int p) ;
```

```
double x_tight() ;
```

```
svg_boxplot & x_tight(double tight) ;
```

```
unsigned int x_value_font_size() ;
```

```
svg_boxplot & x_value_font_size(unsigned int i) ;
```

```
int x_value_ioflags() ;
```

```
svg_boxplot & x_value_ioflags(int flags) ;
```

```
int x_value_precision() ;
```

```
svg_boxplot & x_value_precision(int digits) ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
svg_color x_values_color() ;
```

```
svg_boxplot & x_values_color(const svg_color & col) ;
```

```
const std::string & x_values_font_family() ;
```

```
svg_boxplot & x_values_font_family(const std::string & family) ;
```

```
unsigned int x_values_font_size() ;
```

```
svg_boxplot & x_values_font_size(unsigned int i) ;
```

```
std::ios_base::fmtflags x_values_ioflags() ;
```

```
svg_boxplot & x_values_ioflags(std::ios_base::fmtflags f) ;
```

```
bool x_values_on() ;
```

```
svg_boxplot & x_values_on(bool b) ;
```

```
int x_values_precision() ;
```

```
svg_boxplot & x_values_precision(int p) ;
```

```
int x_values_rotation() ;
```

```
svg_boxplot & x_values_rotation(rotate_style rotate) ;
```

```
bool x_with_zero() ;
```

```
svg_boxplot & x_with_zero(bool b) ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
template<class T > svg_boxplot & y_autoscale(const T & container) ;
```

Set whole data container to use to calculate autoscaled values.

Returns: reference to svg_boxplot to make chainable.

```
template<class T > svg_boxplot & y_autoscale(const T & begin, const T & end) ;
```

Set container and iterator range to use to calculate autoscaled values.

Returns: reference to svg_boxplot to make chainable.

```
svg_boxplot & y_autoscale(std::pair< double, double > p) ;
```

Set Y min & max pair values to use for autoscale.

Returns: reference to svg_boxplot to make chainable.

```
svg_boxplot & y_autoscale(double first, double second) ;
```

Set Y min & max pair values to use for autoscale.

Returns: If to use y_autoscaled values.
Returns: reference to svg_boxplot to make chainable.

```
svg_boxplot & y_autoscale(bool b) ;
```

Set true if Y axis to use autoscale.

Returns: If to use y_autoscaled values.
Returns: reference to svg_boxplot to make chainable.

```
bool y_autoscale() ;
```

Returns: true if Y axis to use autoscale.

```
svg_color y_axis_color() ;
```

```
svg_boxplot & y_axis_color(const svg_color & col) ;
```

```
bool y_axis_on() ;
```

```
svg_boxplot & y_axis_on(bool is) ;
```

```
double y_axis_position() ;
```

Returns: Position of the vertical Y-axis line (on the border).

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
svg_boxplot & y_axis_position(int pos) ;
```

Set position of the vertical Y-axis line (on the border). But controlled by the intersection with X-axis, so this only changes the default position from bottom to top, but will be changed if X-axis intersects the X-axis (that is if X-axis includes zero).

Returns: reference to svg_boxplot to make chainable.

```
std::string y_label() ;
```

```
svg_boxplot & y_label(const std::string & str) ;
```

Set Y axis label.

Returns: reference to svg_boxplot to make chainable.

```
svg_color y_label_color() ;
```

```
svg_boxplot & y_label_color(const svg_color & col) ;
```

Set font color for Y axis label.

Returns: reference to svg_boxplot to make chainable.

```
svg_boxplot & y_label_font_size(unsigned int size) ;
```

Set font size for Y axis label.

Returns: reference to svg_boxplot to make chainable.

```
svg_boxplot & y_label_on(bool cmd) ;
```

Set true if Y axis name or label, for example: "width of thing".

Returns: reference to svg_boxplot to make chainable.

```
std::string y_label_text() ;
```

Returns: Text of label for Y axis.

```
std::string y_label_units() ;
```

```
svg_boxplot & y_label_units(const std::string & str) ;
```

```
bool y_labels_strip_e0s() ;
```

```
svg_boxplot & y_major_interval(double inter) ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Set the interval between major ticks on the Y axis.

```
svg_boxplot & y_major_labels_on(int cmd) ;
```

Set direction of Y major labels. < 0 means to left (default), 0 (false) means none, > 0 means to right.

Returns: reference to svg_boxplot to make chainable.

```
svg_boxplot & y_major_tick_color(const svg_color & col) ;
```

Set Y major ticks color.

Returns: reference to svg_boxplot to make chainable.

```
svg_boxplot & y_major_tick_length(unsigned int length) ;
```

Set the length of major ticks on the Y axis.

Returns: reference to svg_boxplot to make chainable.

```
svg_boxplot & y_major_tick_width(unsigned int width) ;
```

Set the width of major ticks on the Y axis.

Returns: reference to svg_boxplot to make chainable.

```
double y_minor_interval() ;
```

```
svg_boxplot & y_minor_tick_color(const svg_color & col) ;
```

Set Y minor ticks color.

Returns: reference to svg_boxplot to make chainable.

```
svg_boxplot & y_minor_tick_length(unsigned int length) ;
```

Set the length of minor ticks on the Y axis.

Returns: reference to svg_boxplot to make chainable.

```
svg_boxplot & y_minor_tick_width(unsigned int width) ;
```

Set the width of minor ticks on the Y axis.

Returns: reference to svg_boxplot to make chainable.

```
svg_boxplot & y_num_minor_ticks(unsigned int num) ;
```

Set the number of minor ticks between major ticks on the Y axis. For example, 1 gives alternating major and minor ticks, 4 is more useful giving major ticks at 1, 5, 10, 15... 9 gives major ticks at 10, 20, 30...

Returns: reference to svg_boxplot to make chainable.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

38.
```
std::pair< double, double > y_range() ;
```

Set y_range using a pair of doubles (and do not use autoscale).

39.
```
svg_boxplot & y_range(double min_y, double max_y) ;
```

Set range of Y values for Y axis (and do not use autoscale).

Returns: reference to svg_boxplot to make chainable.

`svg_boxplot` protected member functions

1.
```
void adjust_limits(double & x, double & y) ;
```

If value of a data point reaches limit of max, min, infinity, use the appropriate plot min or max value instead.

2.
```
void clear_background() ;
```

Clear the whole image background layer of the SVG plot.

3.
```
void clear_grids() ;
```

Clear the grids layer of the SVG plot.

4.
```
void clear_legend() ;
```

Clear the legend layer of the SVG plot.

5.
```
void clear_plot_background() ;
```

Clear the plot area background layer of the SVG plot.

6.
```
void clear_points() ;
```

Clear the data points layer of the SVG plot.

7.
```
void clear_title() ;
```

Clear the plot title layer of the SVG plot.

8.
```
void clear_x_axis() ;
```

Clear the X axis layer of the SVG plot.

9.
```
void clear_y_axis() ;
```

Clear the Y axis layer of the SVG plot.

10.
```
void draw_legend() ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Draw the legend border, text header (if any) and marker lines and/or shapes.

11.
```
void draw_plot_point(double x, double y, g_element & g_ptr,
                     plot_point_style & sty, unc ux, unc uy) ;
```

Draw a plot data point marker shape whose size and stroke and fill colors are specified in plot_point_style.

12.
```
void draw_plot_point_value(double x, double y, g_element & g_ptr,
                           value_style & val_style,
                           plot_point_style & point_style, unc uvalue) ;
```

Write one data point (X or Y) value as a string, for example "1.23e-2", near the data point marker. Unecessary e, +, & leading exponent zeros may optionally be stripped, and the position and rotation controlled. Uncertainty estimate, typically standard deviation (half conventional 95% confidence 'plus or minus') may be optionally be appended. Degrees of freedom estimate (number of replicates) may optionally be appended. For example: "3.45 +-0.1(10)" The precision and format (scientific, fixed), and color and font type and size can be controlled too.

Unicode space plusminus glyph.

13.
```
void draw_plot_point_values(double x, double y, g_element & x_g_ptr,
                            g_element & y_g_ptr, const value_style & x_sty,
                            const value_style & y_sty, unc uncx, unc uncy) ;
```

Write the **pair** of data point's X and Y values as a string. If a separator, then both on the same line, for example "1.23, 3.45", or "[5.6, 7.8] X value_style is used to provide the prefix and separator, and Y value_style to provide the suffix. For example, x_style prefix("[X=", and separator ",\<ampersand\>\#x00A0;Y= ", " and Y value_style = "]" will produce a value label like "[X=-1.23, Y=4.56]" Note the need to use a Unicode space <ampersand>#x00A0; for get space for all browsers. For as long a string as this you may need to make the total image size bigger, and to orient the value labels with care. draw_plot_point_values is only when both x and Y pairs are wanted.

Also strip unnecessary e, + and leading exponent zeros, if required.

Unicode space plusminus glyph.

14.
```
void draw_x_major_tick(double value, path_element & tick_path,
                       path_element & grid_path) ;
```

Draw major ticks - and grid too if wanted. If major_value_labels_side then value shown beside the major tick.

15.
```
void draw_x_minor_tick(double value, path_element & tick_path,
                       path_element & grid_path) ;
```

Draw X-axis minor ticks, and optional grid. (Value is NOT (yet) shown beside the minor tick).

16.
```
void place_legend_box() ;
```

Place legend box (if required).

17.
```
void size_legend_box() ;
```

Calculate how big the legend box needs to be.

18.
```
std::string sv(double v, const value_style & sty, bool precise = false) ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Strip from double value any unecessary e, +, & leading exponent zeros, reducing "1.200000" to "1.2" or "3.4e1"...

19.
```
void transform_point(double & x, double & y) ;
```

Scale & shift both X & Y to graph Cartesian coordinates.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Class svg_boxplot_series

boost::svg::svg_boxplot_series — Information about a data series to be box plotted.

Synopsis

```
// In header: <boost/svg_plot/svg_boxplot.hpp>


class svg_boxplot_series {
public:
  // construct/copy/destruct
  template<class T >
    svg_boxplot_series(T, T, const std::string &, double, svg_style,
                       svg_style, svg_style, double, svg_style, svg_style,
                       plot_point_style, plot_point_style, int, value_style,
                       text_style);

  // public member functions
  svg_color axis_color() ;
  svg_boxplot_series & axis_color(const svg_color &) ;
  double axis_width() ;
  svg_boxplot_series & axis_width(double) ;
  svg_color box_border() ;
  svg_boxplot_series & box_border(const svg_color &) ;
  svg_color box_fill() ;
  svg_boxplot_series & box_fill(const svg_color &) ;
  svg_boxplot_series & box_style(svg_style &) ;
  svg_style & box_style() ;
  double box_width() ;
  svg_boxplot_series & box_width(double) ;
  void calculate_quantiles() ;
  svg_color extreme_outlier_color() ;
  svg_boxplot_series & extreme_outlier_color(const svg_color &) ;
  svg_color extreme_outlier_fill() ;
  svg_boxplot_series & extreme_outlier_fill(const svg_color &) ;
  point_shape extreme_outlier_shape() ;
  svg_boxplot_series & extreme_outlier_shape(point_shape) ;
  int extreme_outlier_size() ;
  svg_boxplot_series & extreme_outlier_size(int) ;
  svg_color max_whisker_color() ;
  svg_boxplot_series & max_whisker_color(const svg_color &) ;
  double max_whisker_width() ;
  svg_boxplot_series & max_whisker_width(double) ;
  svg_color median_color() ;
  svg_boxplot_series & median_color(const svg_color &) ;
  svg_boxplot_series & median_style(svg_style &) ;
  svg_style & median_style() ;
  double median_width() ;
  svg_boxplot_series & median_width(double) ;
  svg_color min_whisker_color() ;
  svg_boxplot_series & min_whisker_color(const svg_color &) ;
  double min_whisker_width() ;
  svg_boxplot_series & min_whisker_width(double) ;
  svg_color outlier_color() ;
  svg_boxplot_series & outlier_color(const svg_color &) ;
  svg_color outlier_fill() ;
  svg_boxplot_series & outlier_fill(const svg_color &) ;
  point_shape outlier_shape() ;
  svg_boxplot_series & outlier_shape(point_shape) ;
  int outlier_size() ;
  svg_boxplot_series & outlier_size(int) ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
  svg_boxplot_series & outlier_style(plot_point_style &) ;
  plot_point_style & outlier_style() ;
  int quartile_definition() ;
  svg_boxplot_series & quartile_definition(int) ;
  const std::string title() ;
  svg_boxplot_series & title(const std::string &) ;
  double whisker_length() ;
  svg_boxplot_series & whisker_length(double) ;
  svg_style axis_style_;  // line widths and colors of X and Y axes.
  svg_style box_style_;  // line widths and colors of box.
  double box_width_;  // Width of boxplot box.
  plot_point_style ext_outlier_;  // Style (shape, color...) for marking 'extreme' outliers.
  bool extreme_outlier_values_on_;  // True if extreme outliers are to have their values labelled.
  std::vector< double > extreme_outliers_;  // Any data values that are judged extreme outliers.
  svg_style max_whisker_style_;  // Color and width etc of boxplot minimum 'whisker'.
  double median_;  // 2nd middle quartile.
  svg_style median_style_;  // line widths and colors of median marker.
  plot_point_style mild_outlier_;  // Style (shape, color...) for marking 'mild' outliers.
  svg_style min_whisker_style_;  // Color and width etc of boxplot minimum 'whisker'.
  bool outlier_values_on_;  // True if mild outliers are to have their values labelled.
  std::vector< double > outliers_;  // Any data values that are judged outliers.
  double q1_;  // 1st lower quartile.
  double q3_;  // 3rd upper quartile.
  int quartile_definition_;  // The definition of the quartile can be selected.
  std::vector< double > series_;  // Data series for the boxplot.
  text_element series_info_;  // information about the data series.
  text_style series_style_;  // Style (font etc) for text.
  double text_margin_;  // Margin (SVG units, default pixels) around text items.
  value_style values_style_;  // Style for data value labels.
  double whisker_length_;  // Length of boxplot 'whisker'.
  double whisker_max_;  // Maximum of whisker.
  double whisker_min_;  // Minimum of whisker.
};
```

Description

A Box Plot that can contain several boxplot data series. Median, whiskers and outliers are computed for each series.

http://en.wikipedia.org/wiki/Boxplot

`svg_boxplot_series` public construct/copy/destruct

1.
```
template<class T >
  svg_boxplot_series(T begin, T end, const std::string & title, double bw,
                     svg_style bs, svg_style ms, svg_style as, double wl,
                     svg_style minws, svg_style maxws, plot_point_style os,
                     plot_point_style extos, int q_def, value_style vs,
                     text_style ss);
```

All other parameters can also be added using chainable functions.

Default Constructor. std::vector copy is sorted and used for fast lookup of quartile values. std::vector<double> series(begin, end); now a member

Not enough data values to usefully calculate quartiles.

`svg_boxplot_series` public member functions

1.
```
svg_color axis_color() ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Color of vertical axis whisker line in box.

Returns: Color of axis line.

2.
```
svg_boxplot_series & axis_color(const svg_color & color) ;
```

Width of median line in box.

Set color of axis line.

Returns: reference to svg_boxplot_series to make chainable.

3.
```
double axis_width() ;
```

Width of vertical axis whisker line in box.

Returns: width of axis line.

4.
```
svg_boxplot_series & axis_width(double l) ;
```

Color of vertical axis whisker line in box.

Set width of axis line.

Returns: reference to svg_boxplot_series to make chainable.

5.
```
svg_color box_border() ;
```

Color of border (not fill) to box.

Returns: color of box border.

6.
```
svg_boxplot_series & box_border(const svg_color & color) ;
```

Color of box fill, not border.

Set color of box border.

Returns: reference to svg_boxplot_series to make chainable.

7.
```
svg_color box_fill() ;
```

Color of box fill, not border.

Returns: color of box fill.

8.
```
svg_boxplot_series & box_fill(const svg_color & color) ;
```

Width of whisker line.

Set color of box fill (not border).

Returns: reference to svg_boxplot_series to make chainable.

9.
```
svg_boxplot_series & box_style(svg_style & bs) ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Set entire box style.

Returns: reference to style of box.
Returns: reference to svg_boxplot_series to make chainable.

10.
```
svg_style & box_style() ;
```

Returns: width of vertical axis whisker line in box.
Returns: box style.

11.
```
double box_width() ;
```

Set width of box.

Returns: width of the box.

12.
```
svg_boxplot_series & box_width(double l) ;
```

Set width of the box.

Returns: length of whisker line.
Returns: reference to svg_boxplot_series to make chainable.

13.
```
void calculate_quantiles() ;
```

Divide sorted data set into four equal parts, called quartiles, so each part represent 1/4th of the sampled population.

Michael Frigge, David C. Hoaglin and Boris Iglewicz The American Statistician, Vol. 43, No. 1 (Feb., 1989), pp. 50-54 Tukey, J. W. Exploratory Data Analysis, Addison Wesley (1977, p 33)

"Some Implementations of the Boxplot"

x[1] .. x[n] == series[0] ... series[n - 1] q1_ = (1 - g) * x[j] + g * [j+1];

Fences (beyond which lie outliers) are at q1 - k * (q3 - q1) and q3 - k * (q3 - q1) commonly k = 1.5, but can be 2. Extreme outlier usually use k = 3.

Interquartile range.

Separate any minimum outliers and extreme outliers.

Separate any maximum outliers and extreme outliers.

14.
```
svg_color extreme_outlier_color() ;
```

Set color of extreme outlier marker.

Returns: fill color of extreme outlier line in box.

15.
```
svg_boxplot_series & extreme_outlier_color(const svg_color & color) ;
```

Set fill color of extreme outlier line in box.

Returns: fill color of outlier marker.
Returns: reference to svg_boxplot_series to make chainable.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

16.
```
svg_color extreme_outlier_fill() ;
```

Set color of extreme outlier marker.

Returns: fill color of extreme outlier line in box.

17.
```
svg_boxplot_series & extreme_outlier_fill(const svg_color & color) ;
```

Set fill color of extreme outlier line in box.

Returns: color of extreme outlier marker.
Returns: reference to svg_boxplot_series to make chainable.

18.
```
point_shape extreme_outlier_shape() ;
```

Shape of extreme outlier marker. !

Returns: shape of extreme outlier marker.

19.
```
svg_boxplot_series & extreme_outlier_shape(point_shape shape) ;
```

Set shape of extreme outlier marker.

Returns: size of outlier marker.
Returns: reference to svg_boxplot_series to make chainable.

20.
```
int extreme_outlier_size() ;
```

Size of extreme outlier marker.

Returns: Size of extreme outlier marker.

21.
```
svg_boxplot_series & extreme_outlier_size(int size) ;
```

Set size of extreme outlier marker.

Returns: shape of extreme outlier marker.
Returns: reference to svg_boxplot_series to make chainable..

22.
```
svg_color max_whisker_color() ;
```

Returns: color of maximum whisker.

23.
```
svg_boxplot_series & max_whisker_color(const svg_color & col) ;
```

Set color of minimum whisker.

Returns: width of whisker line.
Returns: reference to svg_boxplot_series to make chainable.

24.
```
double max_whisker_width() ;
```

Width of whisker line.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Returns: line width of maximum whisker.

25.
```
svg_boxplot_series & max_whisker_width(double l) ;
```

Color of whisker.

Returns: line width of maximum whisker.

reference to svg_boxplot_series to make chainable.

26.
```
svg_color median_color() ;
```

Color of median line in box.

Returns: color of median line in box.

27.
```
svg_boxplot_series & median_color(const svg_color & color) ;
```

Color of border (not fill) to box.

Set color of median line in box.

Returns: reference to svg_boxplot_series to make chainable.

28.
```
svg_boxplot_series & median_style(svg_style & ms) ;
```

Set entire median style.

Returns: reference to style of median marker line.
Returns: reference to svg_boxplot_series to make chainable.

29.
```
svg_style & median_style() ;
```

Set reference to style of box.

Returns: median style.

30.
```
double median_width() ;
```

Width of median line in box.

Returns: width of median line in box.

31.
```
svg_boxplot_series & median_width(double l) ;
```

Color of median line in box.

Set width of median line in box.

Returns: reference to svg_boxplot_series to make chainable..

32.
```
svg_color min_whisker_color() ;
```

Color of boxplot whisker.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Returns: color of minimum whisker.

33.
```
svg_boxplot_series & min_whisker_color(const svg_color & col) ;
```

Set color of minimum whisker.

Returns: width of box.
Returns: reference to svg_boxplot_series to make chainable.

34.
```
double min_whisker_width() ;
```

Set width of whisker line.

Returns: line width of minimum whisker.

35.
```
svg_boxplot_series & min_whisker_width(double l) ;
```

Set line width of minimum whisker.

Returns: color of boxplot whisker.
Returns: reference to svg_boxplot_series to make chainable..

36.
```
svg_color outlier_color() ;
```

Set color of outlier marker.

Returns: color of outlier line in box.

37.
```
svg_boxplot_series & outlier_color(const svg_color & color) ;
```

Set reference to style of outlier marker line.

Set color of outlier line in box.

Returns: reference to svg_boxplot_series to make chainable.

38.
```
svg_color outlier_fill() ;
```

Set fill color of outlier marker.

Returns: fill color of mild outlier line in box.

39.
```
svg_boxplot_series & outlier_fill(const svg_color & color) ;
```

Set fill color of mild outlier line in box.

Returns: color of outlier marker.
Returns: reference to svg_boxplot_series to make chainable.

40.
```
point_shape outlier_shape() ;
```

Set shape of outlier marker.

Returns: shape of outlier marker.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

41.
```
svg_boxplot_series & outlier_shape(point_shape shape) ;
```

Set shape of outlier marker.

Returns: color of extreme outlier marker.
Returns: reference to svg_boxplot_series to make chainable.

42.
```
int outlier_size() ;
```

Set size of outlier marker.

Returns: Size of outlier marker.

43.
```
svg_boxplot_series & outlier_size(int size) ;
```

Set size of outlier marker.

Returns: shape of outlier marker.
Returns: reference to svg_boxplot_series to make chainable.

44.
```
svg_boxplot_series & outlier_style(plot_point_style & os) ;
```

Set entire outlier style.

Returns: reference to style of outlier marker line.
Returns: reference to svg_boxplot_series to make chainable.

45.
```
plot_point_style & outlier_style() ;
```

Returns: outlier_style.

46.
```
int quartile_definition() ;
```

Set H&F quartile definition.

Returns: Choice of H&F quartile definition.

47.
```
svg_boxplot_series & quartile_definition(int def) ;
```

set Choice of H&F quartile definition.

Returns: size of extreme outlier marker.
Returns: reference to svg_boxplot_series to make chainable.

48.
```
const std::string title() ;
```

Set title for data series.

Returns: title of a data series.

49.
```
svg_boxplot_series & title(const std::string & t) ;
```

Set title of a data series.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Returns: reference to svg_boxplot_series to make chainable.

50.
```
double whisker_length() ;
```

Set length of whisker line.

Returns: minimum and maximum whisker length.

51.
```
svg_boxplot_series & whisker_length(double l) ;
```

Set minimum and maximum whisker length.

Returns: title for boxplot.
Returns: reference to svg_boxplot_series to make chainable.

Header <boost/svg_plot/svg_color.hpp>

SVG standard names of colors, and functions to create and output colors.

9 Feb 2009

Jacob Voytko & Paul A. Bristow

```
namespace boost {
  namespace svg {
    class svg_color;

    enum svg_color_constant;

    svg_color color_array;    // SVG standard colors,.
    svg_color constant_to_rgb(svg_color_constant);
    void constant_to_rgb(svg_color_constant, unsigned char &, unsigned char &,
                         unsigned char &);
    bool is_blank(const svg_color &);
    bool operator!=(const svg_color &, const svg_color &);
    std::ostream & operator<<(std::ostream &, const svg_color &);
    bool operator==(const svg_color &, const svg_color &);
  }
}
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Class svg_color

boost::svg::svg_color — SVG standard colors, see also enum svg_color_constant.

Synopsis

```
// In header: <boost/svg_plot/svg_color.hpp>


class svg_color {
public:
  // construct/copy/destruct
  svg_color(svg_color_constant);
  svg_color(bool);
  svg_color(int, int, int);

  // public member functions
  unsigned int blue() const;
  unsigned int green() const;
  bool is_blank() const;
  bool operator!=(const svg_color &) ;
  bool operator==(const svg_color &) ;
  unsigned int red() const;
  void write(std::ostream &) ;
  unsigned char b_;  // blue unsigned char provides range [0 to 255].
  unsigned char g_;  // green unsigned char provides range [0 to 255].
  bool is_blank_;  // true means "Not to be displayed" a 'pseudo-color'. If is_blank_ == true ↵
should write output to SVG XML file as "none".
  unsigned char r_;  // red unsigned char provides range [0 to 255].
};
```

Description

svg_color is the struct that contains information about RGB colors. For the constructor, the SVG standard specifies that numbers outside the normal rgb range are to be accepted, but are constrained to acceptable range of integer values [0, 255].

`svg_color` public construct/copy/destruct

1. ```
   svg_color(svg_color_constant col);
   ```

   Set a color (including blank) using the SVG 'standard' colors defined in enum svg_color_constant

2. ```
   svg_color(bool is);
   ```

   Constructor from bool permits svg_color my_blank(false) as a (non-)color.

   with same effect as svg_color my_blank(blank); svg_color(true) means default (black?) svg_color(false) means blank. For example: plot.area_fill(false) will be a blank == no fill. plot.area_fill(true) will be a default(black) fill.

3. ```
   svg_color(int red, int green, int blue);
   ```

   Construct an SVG color from RGB values. Constrain rgb to [0 .. 255]. Default is to construct a 'pseudo-color' blank.

`svg_color` public member functions

1. ```
   unsigned int blue() const;
   ```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

return blue component of color [0, 255]

2. ```
   unsigned int green() const;
   ```

   return green component of color [0, 255]

3. ```
   bool is_blank() const;
   ```

   Returns: true if color is blank.

4. ```
   bool operator!=(const svg_color & rhs) ;
   ```

   Compare colors (for not equal).

5. ```
   bool operator==(const svg_color & rhs) ;
   ```

   Compare colors (for equal).

6. ```
   unsigned int red() const;
   ```

   return red component of color [0, 255]

7. ```
   void write(std::ostream & os) ;
   ```

   Write to ostream a color in svg format. Usage: my_color.write(cout); Outputs: rgb(127,255,212)

   Note lower case (whereas operator<< uses uppercase).

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Type svg_color_constant

boost::svg::svg_color_constant — Colors that have SVG standard special names.

Synopsis

```
// In header: <boost/svg_plot/svg_color.hpp>



enum svg_color_constant { aliceblue, antiquewhite, aqua, aquamarine, azure,
                          beige, bisque, black, blanchedalmond, blue,
                          blueviolet, brown, burlywood, cadetblue, chartreuse,
                          chocolate, coral, cornflowerblue, cornsilk, crimson,
                          cyan, darkblue, darkcyan, darkgoldenrod, darkgray,
                          darkgreen, darkgrey, darkkhaki, darkmagenta,
                          darkolivegreen, darkorange, darkorchid, darkred,
                          darksalmon, darkseagreen, darkslateblue,
                          darkslategray, darkslategrey, darkturquoise,
                          darkviolet, deeppink, deepskyblue, dimgray, dimgrey,
                          dodgerblue, firebrick, floralwhite, forestgreen,
                          fuchsia, gainsboro, ghostwhite, gold, goldenrod,
                          gray, grey, green, greenyellow, honeydew, hotpink,
                          indianred, indigo, ivory, khaki, lavender,
                          lavenderblush, lawngreen, lemonchiffon, lightblue,
                          lightcoral, lightcyan, lightgoldenrodyellow,
                          lightgray, lightgreen, lightgrey, lightpink,
                          lightsalmon, lightseagreen, lightskyblue,
                          lightslategray, lightslategrey, lightsteelblue,
                          lightyellow, lime, limegreen, linen, magenta,
                          maroon, mediumaquamarine, mediumblue, mediumorchid,
                          mediumpurple, mediumseagreen, mediumslateblue,
                          mediumspringgreen, mediumturquoise, mediumvioletred,
                          midnightblue, mintcream, mistyrose, moccasin,
                          navajowhite, navy, oldlace, olive, olivedrab,
                          orange, orangered, orchid, palegoldenrod, palegreen,
                          paleturquoise, palevioletred, papayawhip, peachpuff,
                          peru, pink, plum, powderblue, purple, red,
                          rosybrown, royalblue, saddlebrown, salmon,
                          sandybrown, seagreen, seashell, sienna, silver,
                          skyblue, slateblue, slategray, slategrey, snow,
                          springgreen, steelblue, tanned, teal, thistle,
                          tomato, turquoise, violet, wheat, white, whitesmoke,
                          yellow, yellowgreen, blank };
```

Description

SVG standard names for some colors. See http://www.w3.org/TR/SVG/types.html#ColorKeywords.

The reason that the underscore separator convention does not match the normal Boost format is that these names that are specified by the SVG standard. http://www.w3.org/TR/SVG/types.html#ColorKeywords color "tan" is also renamed to "tanned" to avoid clash with global function name tan in math.h.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Global color_array

boost::svg::color_array — SVG standard colors,.

Synopsis

```
// In header: <boost/svg_plot/svg_color.hpp>

svg_color color_array;
```

Description

svg_color_constant

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Function constant_to_rgb

boost::svg::constant_to_rgb

Synopsis

```
// In header: <boost/svg_plot/svg_color.hpp>


svg_color constant_to_rgb(svg_color_constant c);
```

Description

Convert a svg color constant enum svg_color_constant to a svg_color.

Returns: svg_color Example: constant_to_rgb(4) or constant_to_rgb(aquamarine) gives svg_color(127, 255, 212) // aquamarine.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Function constant_to_rgb

boost::svg::constant_to_rgb

Synopsis

```
// In header: <boost/svg_plot/svg_color.hpp>


void constant_to_rgb(svg_color_constant c, unsigned char & r,
                     unsigned char & g, unsigned char & b);
```

Description

Convert a named SVG standard color, see enum svg_color_constant to update three variables (r, g, b) holding red, green and blue values. Assumes is c NOT the blank color, and asserts if it is. This comment does not appear - for reasons unclear.

Convert a named SVG standard color, see enum svg_color_constant to update three variables (r, g, b) holding red, green and blue values. Assumes is c NOT the blank color, and asserts if it is. This comment does not appear - for reasons unclear.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Function is_blank

boost::svg::is_blank

Synopsis

```
// In header: <boost/svg_plot/svg_color.hpp>


bool is_blank(const svg_color & col);
```

Description

Returns: true if color is blank.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Function operator!=

boost::svg::operator!=

Synopsis

```
// In header: <boost/svg_plot/svg_color.hpp>


bool operator!=(const svg_color & lhs, const svg_color & rhs);
```

Description

Compare colors (for not equal).

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Function operator<<

boost::svg::operator<<

Synopsis

```
// In header: <boost/svg_plot/svg_color.hpp>


std::ostream & operator<<(std::ostream &, const svg_color &);
```

Description

Output color to stream as RGB. See svg_color_constant for example: "RGB(138, 43 , 226)" for blueviolet. This comment does not appear - for reasons entirely unclear.

Usage: svg_color my_color(127, 255, 212); cout << "my_color " << my_color << endl; Outputs: my_color RGB(127,255,212) cout << "magenta " << svg_color(magenta) << endl; but caution! cout << magenta << endl; outputs 85 because magenta is an enum svg_color_constant !

Output color to stream as RGB. See svg_color_constant for example: "RGB(138, 43 , 226)" for blueviolet. This comment does not appear - for reasons entirely unclear.

Usage: svg_color my_color(127, 255, 212); cout << "my_color " << my_color << endl; Outputs: my_color RGB(127,255,212) cout << "magenta " << svg_color(magenta) << endl; but caution! cout << magenta << endl; outputs 85 because magenta is an enum svg_color_constant !

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Function operator==

boost::svg::operator==

Synopsis

```
// In header: <boost/svg_plot/svg_color.hpp>


bool operator==(const svg_color & lhs, const svg_color & rhs);
```

Description

Compare colors (for equal).

Header <boost/svg_plot/svg_fwd.hpp>

Forward declarations of SVG classes and functions.

Jacob Voytko & Paul A. Bristow
Doxygen descriptions are in the corresponding .hpp files.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
namespace boost {
  namespace svg {
    const std::string author();
    svg & author(const std::string);
    bool boost_license();
    svg & boost_license(bool);
    svg & circle(double x, double y, unsigned int radius);
    void clip_id(const std::string & id);
    int coord_precison();
    svg & coord_precison(int);
    const std::string copyright_date();
    svg & copyright_date(const std::string d);
    const std::string copyright_holder();
    svg & copyright_holder(const std::string d);
    std::string & description();
    svg & description(const std::string d);
    const std::string document_title();
    svg & document_title(const std::string d);
    svg & ellipse(double rx, double ry, double cx, double cy);
    svg_style & fill_color(const svg_color & col);
    svg_color fill_color();
    bool fill_on();
    std::string id();
    void id(const std::string & id);
    bool is_license();
    svg & license(bool);
    svg & license(std::string repro, std::string distrib, std::string attrib,
                  std::string commercial);
    svg & line(double x1, double y1, double x2, double y2);
    svg & load_stylesheet(const std::string & input);
    svg & rect(double x1, double y1, double x2, double y2);
    svg & size(unsigned int x, unsigned int y);
    svg_style & stroke_color(const svg_color & col);
    svg_color stroke_color();
    bool stroke_on();
    svg_style & stroke_width(unsigned int width);
    unsigned int stroke_width();
    svg_style & style();
    svg & text(double x, double y, const std::string & text, int text_size,
               const std::string & font, const std::string & style,
               const std::string & weight, const std::string & stretch,
               const std::string & decoration, align_style align, int rotate);
    int width_on();
    svg & write(const std::string & file);
    svg & write(std::ostream & s_out);
    unsigned int x_size();
    svg & x_size(unsigned int x);
    unsigned int y_size();
    svg & y_size(unsigned int y);
  }
}
```

Header <boost/svg_plot/svg_style.hpp>

Styles for SVG specifying font, sizes, shape, color etc for text, values, lines, axes etc.

SVG style information is fill, stroke, width, line & bezier curve. This module provides struct plot_point_style & struct plot_line_style and class svg_style holding the styles. See http://www.w3.org/TR/SVG11/styling.html

Mar 2009

Jacob Voytko and Paul A. Bristow

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
namespace boost {
  namespace svg {
    class axis_line_style;
    class bar_style;
    class box_style;
    class histogram_style;
    class plot_line_style;
    class plot_point_style;
    class svg_style;
    class text_style;
    class ticks_labels_style;
    class value_style;

    // Options for bar to draw bar charts.
    enum bar_option { y_block =  -2, y_stick =  -1, no_bar =  0,
                      x_stick =  +1, x_block =  +2 };

    // dimension of plot. (Used so that an axis knows what type it is, or none = N).
    enum dim { N =  0, X =  1, Y =  2 };

    // options for histograms.
    enum histogram_option { no_histogram =  0, column =  +1 };

    // used for marking a data point.
    enum point_shape { none =  0, round, square, point, egg, unc_ellipse,
                       vertical_line, horizontal_line, vertical_tick,
                       horizontal_tick, cone, triangle, star, lozenge,
                       diamond, heart, club, spade, asterisk, x, cross,
                       symbol };

    // Rotation in degrees clockwise from horizontal.
    enum rotate_style { horizontal =  0, slopeup =  -30, uphill =  -45,
                        steepup =  -60, upward =  -90, backup =  -135,
                        leftward =  -180, rightward =  360,
                        slopedownhill =  30, downhill =  45, steepdown =  60,
                        downward =  90, backdown =  135, upsidedown =  180 };

    text_style no_style;    // Text style that uses all constructor defaults.
    static const double wh;    // font text width/height ratio.

    // plot_point_style that uses all the defaults.
    plot_point_style default_plot_point_style();
    bool operator!=(const text_style &, const text_style &);
    std::ostream & operator<<(std::ostream &, plot_line_style);
    std::ostream & operator<<(std::ostream &, plot_point_style);
    std::ostream & operator<<(std::ostream &, const text_style &);
    std::ostream & operator<<(std::ostream &, const svg_style &);
    bool operator==(const text_style &, const text_style &);
    double string_svg_length(const std::string &, const text_style &);
  }
}
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Class axis_line_style

boost::svg::axis_line_style — Style of the X or Y-axes lines.

Synopsis

```
// In header: <boost/svg_plot/svg_style.hpp>


class axis_line_style {
public:
  // construct/copy/destruct
  axis_line_style(dim = X, double = -10., double = +10.,
                  const svg_color = black, double = 1, int = 0, bool = true,
                  bool = false, bool = true, double = -1);

  // public member functions
  bool axis_line_on() const;
  svg_color color() ;
  axis_line_style & color(const svg_color &) ;
  axis_line_style & label_on(bool) ;
  bool label_on() const;
  axis_line_style & label_units_on(bool) ;
  bool label_units_on() const;
  double position() ;
  axis_line_style & position(int) ;
  double width() ;
  axis_line_style & width(double) ;
  double axis_;  // Depending on value of dim, either X-axis (y = 0) transformed into SVG Y co↵
ordinates or Y-axis (x = 0) transformed into SVG X coordinates (-1 if not calculated yet).
  bool axis_line_on_;  // Draw an X horizontal or Y vertical axis line.
  int axis_position_;
  double axis_width_;  // Axis line width.
  svg_color color_;  // Axis line (stroke) color.
  dim dim_;  // None, X or Y.
  bool label_on_;  // Label axis with text - example: "length".
  bool label_units_on_;  // Label axis units, example: "cm".
  double max_;  // maximum Y value (Cartesian units).
  double min_;  // minimum X value (Cartesian units).
};
```

Description

(But NOT the ticks and value labels because different styles for X and Y-axes are possible).

`axis_line_style` public construct/copy/destruct

1.
```
axis_line_style(dim d = X, double min = -10., double max = +10.,
                const svg_color col = black, double width = 1,
                int axis_position = 0, bool label_on = true,
                bool label_units_on = false, bool axis_lines_on = true,
                double axis = -1);
```

`axis_line_style` public member functions

1.
```
bool axis_line_on() const;
```

If returns true, then either an X or a Y axis line to be drawn.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

2.
```
svg_color color() ;
```

Returns: color of an axis line.

3.
```
axis_line_style & color(const svg_color & color) ;
```

Set color of an axis line.

Returns: plot_line_style& to make chainable.

4.
```
axis_line_style & label_on(bool is) ;
```

If set true, then axis to be labelled with the label, for example "X axis" (but default "").

Returns: plot_line_style& to make chainable.

5.
```
bool label_on() const;
```

If returns true, then axis to be labelled, for example "X axis".

6.
```
axis_line_style & label_units_on(bool is) ;
```

If set true, then axis to be labelled with the units label, for example " (mm)" (but default "").

Returns: plot_line_style& to make chainable.

7.
```
bool label_units_on() const;
```

If returns true, then axis to be labelled with unit, for example " (mm)"

8.
```
double position() ;
```

Returns: How the axes intersect. enum x_axis_intersect {bottom = -1, x_intersects_y = 0, top = +1}; enum y_axis_intersect {left = -1, y_intersects_x = 0, right = +1}; If axes look like an L, then is bottom left. If a T then y intersects and X is at bottom.

9.
```
axis_line_style & position(int pos) ;
```

How the axes intersect.

Returns: plot_line_style& to make chainable.

10.
```
double width() ;
```

Returns: width of an axis line.

11.
```
axis_line_style & width(double w) ;
```

Set width of an axis line.

Returns: plot_line_style& to make chainable.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Class bar_style

boost::svg::bar_style — Style (color, width, fill) of histogram bars.

Synopsis

```
// In header: <boost/svg_plot/svg_style.hpp>


class bar_style {
public:
  // construct/copy/destruct
  bar_style(const svg_color & = black, const svg_color & = true, double = 2,
            bar_option = no_bar);

  // public member functions
  svg_color & area_fill() ;
  bar_style & area_fill(const svg_color &) ;
  double bar_opt() ;
  bar_style & bar_opt(bar_option) ;
  svg_color & color() ;
  bar_style & color(const svg_color &) ;
  double width() ;
  bar_style & width(double) ;
  svg_color area_fill_;  // Fill color from line to axis.
  bar_option bar_option_;  // stick or bar.
  svg_color color_;  // Color of line (stroke) (no fill color for lines).
  double width_;  // Width of bar, not enclosing line width.
};
```

Description

`bar_style` public construct/copy/destruct

1.
```
bar_style(const svg_color & col = black, const svg_color & acol = true,
          double width = 2, bar_option opt = no_bar);
```

Construct with defaults for all member variables.

Constructor, setting defaults for all member variables.

`bar_style` public member functions

1.
```
svg_color & area_fill() ;
```

Returns: bar rectangle fill color.

2.
```
bar_style & area_fill(const svg_color & f) ;
```

Returns: box_style& to make chainable.

3.
```
double bar_opt() ;
```

Returns: If to use stick or bar for histograms.

4.
```
bar_style & bar_opt(bar_option option) ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Returns: box_style& to make chainable.

5.
```
svg_color & color() ;
```

Returns: Color of bar line or enclosing line.

6.
```
bar_style & color(const svg_color & f) ;
```

Returns: box_style& to make chainable.

7.
```
double width() ;
```

Returns: Width of bar, not enclosing line width.

8.
```
bar_style & width(double w) ;
```

Returns: box_style& to make chainable.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Class box_style

boost::svg::box_style — a rectangular box. (Used for boxplot image and plot window).

Synopsis

```
// In header: <boost/svg_plot/svg_style.hpp>


class box_style {
public:
  // construct/copy/destruct
  box_style(const svg_color & = black, const svg_color & = white, double = 1,
            double = 4., bool = true, bool = false);

  // public member functions
  box_style & border_on(bool) ;
  bool border_on() const;
  svg_color fill() ;
  box_style & fill(const svg_color &) ;
  box_style & fill_on(bool) ;
  bool fill_on() const;
  double margin() ;
  box_style & margin(double) ;
  svg_color stroke() ;
  box_style & stroke(const svg_color &) ;
  double width() ;
  box_style & width(double) ;
  bool border_on_;  // Display the border of the box.
  svg_color fill_;  // Box fill color.
  bool fill_on_;  // Color fill the box.
  double margin_;  // Marginal (pixels) space around the box (inside or out).
  svg_color stroke_;  // Box line (stroke) color.
  double width_;  // plot border rectangle width.
};
```

Description

box_style public construct/copy/destruct

1.
```
box_style(const svg_color & scolor = black, const svg_color & fcolor = white,
          double width = 1, double margin = 4., bool border_on = true,
          bool fill_on = false);
```

Apply fill color.

box_style public member functions

1.
```
box_style & border_on(bool is) ;
```

Set true if the box border should be shown.

Returns: box_style& to make chainable.

2.
```
bool border_on() const;
```

Returns: If the box border should be shown.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

3.
```
svg_color fill() ;
```

Returns: Fill color for box.

4.
```
box_style & fill(const svg_color & color) ;
```

Set fill color for box.

Returns: box_style& to make chainable.

5.
```
box_style & fill_on(bool is) ;
```

Set true if the box should be filled.

6.
```
bool fill_on() const;
```

Returns: if the box should be filled.

7.
```
double margin() ;
```

Returns: marginal (default pixels) space around the box (inside or out).

8.
```
box_style & margin(double w) ;
```

Set marginal (default pixels) space around the box (inside or out).

Returns: box_style& to make chainable.

9.
```
svg_color stroke() ;
```

Returns: (stroke) color for box outline.

10.
```
box_style & stroke(const svg_color & color) ;
```

Set (stroke) color for box outline.

Returns: box_style& to make chainable.

11.
```
double width() ;
```

Returns: width for box.

12.
```
box_style & width(double w) ;
```

Set width for box.

Returns: box_style& to make chainable.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Class histogram_style

boost::svg::histogram_style — Histogram options.

Synopsis

```
// In header: <boost/svg_plot/svg_style.hpp>


class histogram_style {
public:
  // construct/copy/destruct
  histogram_style(histogram_option = no_histogram);

  // public member functions
  double histogram() ;
  histogram_style & histogram(histogram_option) ;
  histogram_option histogram_option_;  // default bar, no_histogram or column.
};
```

Description

`histogram_style` public construct/copy/destruct

1.
```
histogram_style(histogram_option opt = no_histogram);
```

   Set any histogram option.

   Constructor providing defaults for all private data. Line width and area-fill are taken from the plot_line_style style.

`histogram_style` public member functions

1.
```
double histogram() ;
```

2.
```
histogram_style & histogram(histogram_option opt) ;
```

   Set any histogram option.

   Histogram to be shown as sticks or bars.

   Returns: box_style& to make chainable.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Class plot_line_style

boost::svg::plot_line_style — line joining data series values.

Synopsis

```
// In header: <boost/svg_plot/svg_style.hpp>


class plot_line_style {
public:
  // construct/copy/destruct
  plot_line_style(const svg_color & = black, const svg_color & = blank,
                  double = 2, bool = true, bool = false);

  // public member functions
  svg_color & area_fill() ;
  plot_line_style & area_fill(const svg_color &) ;
  plot_line_style & bezier_on(bool) ;
  bool bezier_on() const;
  svg_color & color() ;
  plot_line_style & color(const svg_color &) ;
  plot_line_style & line_on(bool) ;
  bool line_on() const;
  double width() ;
  plot_line_style & width(double) ;
  svg_color area_fill_;  // Fill color from line to axis. == false means color.is_blank = true, ↵
or = blank.
  bool bezier_on_;  // If true, data points will be joined by bezier curved line(s).
  bool line_on_;  // If true, data points will be joined by straight line(s).
  svg_color stroke_color_;  // Stroke color of line. (no fill color for lines).
  double width_;  // Width of line joining data series values.
};
```

Description

`plot_line_style` public construct/copy/destruct

1.
```
plot_line_style(const svg_color & col = black,
                const svg_color & fill_col = blank, double width = 2,
                bool line_on = true, bool bezier_on = false);
```

`plot_line_style` public member functions

1.
```
svg_color & area_fill() ;
```

Returns: if area under line joining data points is to be color filled.

2.
```
plot_line_style & area_fill(const svg_color & f) ;
```

Set if area under line joining data points is to be color filled.

Returns: plot_line_style& to make chainable.

3.
```
plot_line_style & bezier_on(bool is) ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Set true if bezier curved line(s) are to join data points.

Returns: plot_line_style& to make chainable.

4.
```
bool bezier_on() const;
```

Returns: true if bezier curved line(s) are to join data points.

5.
```
svg_color & color() ;
```

Returns: color of line(s) joining data points.

6.
```
plot_line_style & color(const svg_color & f) ;
```

Set color of line(s) joining data points.

Returns: plot_line_style& to make chainable.

7.
```
plot_line_style & line_on(bool is) ;
```

Set true if line(s) are to join data points.

Returns: plot_line_style& to make chainable.

8.
```
bool line_on() const;
```

Returns: True if line(s) will join data points.

9.
```
double width() ;
```

Returns: width of line(s) joining data points.

10.
```
plot_line_style & width(double w) ;
```

Set width of line(s) joining data points.

Returns: plot_line_style& to make chainable.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Class plot_point_style

boost::svg::plot_point_style — Shape, color, of data point markers.

Synopsis

```
// In header: <boost/svg_plot/svg_style.hpp>


class plot_point_style {
public:
  // construct/copy/destruct
  plot_point_style(const svg_color & = black, const svg_color & = blank,
                   int = 5, point_shape = round, const std::string & = "X");

  // public member functions
  svg_color & fill_color() ;
  plot_point_style & fill_color(const svg_color &) ;
  point_shape shape() ;
  plot_point_style & shape(point_shape) ;
  int size() ;
  plot_point_style & size(int) ;
  svg_color & stroke_color() ;
  plot_point_style & stroke_color(const svg_color &) ;
  text_style & style() const;
  plot_point_style & style(text_style) ;
  std::string & symbols() ;
  plot_point_style & symbols(const std::string) ;
  svg_color fill_color_;  // Fill color of the centre of the shape.
  point_shape shape_;  // shape: round, square, point...
  bool show_x_value_;  // If true, show the X value like "1.2" near the point. (If both true, ↵
then show both X and Y as a pair like "1.2, 3.4".).
  bool show_y_value_;  // If true, show the Y value like "3.4" near the point. (If both true, ↵
then show both X and Y as a pair like "1.2, 3.4".).
  int size_;  // Diameter of circle, height of square, font_size ...
  svg_color stroke_color_;  // Color of circumference of shape.
  std::string symbols_;  // Unicode symbol(s) (letters, digits, squiggles etc).
 Caution: not all Unicode symbols are output by all browsers!
 Example: U2721 is Star of David or hexagram, see http://en.wikipedia.org/wiki/Hexagram, sym↵
bols("&#x2721;") Positioning of symbols (especially > 1 symbols) may be inprecise.
  text_style symbols_style_;  // font, size, decoration of symbols.
};
```

Description

(optional value & uncertainty) not implemented yet.

`plot_point_style` public construct/copy/destruct

1.
```
plot_point_style(const svg_color & stroke = black,
                 const svg_color & fill = blank, int size = 5,
                 point_shape shape = round,
                 const std::string & symbols = "X");
```

`plot_point_style` public member functions

1.
```
svg_color & fill_color() ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Returns: fill color of shape or symbol used to mark data value plot point(s).

2.
```
plot_point_style & fill_color(const svg_color & f) ;
```

Set fill color of shape or symbol used to mark data value plot point(s).

Returns: plot_point_style& to make chainable.

3.
```
point_shape shape() ;
```

Returns: shape used to mark data value plot point(s).

4.
```
plot_point_style & shape(point_shape s) ;
```

Set shape used to mark data value plot point(s).

Returns: plot_point_style& to make chainable.

5.
```
int size() ;
```

Returns: size of shape or symbol used to mark data value plot point(s).

6.
```
plot_point_style & size(int i) ;
```

Set size of shape or symbol used to mark data value plot point(s).

Returns: plot_point_style& to make chainable.

7.
```
svg_color & stroke_color() ;
```

Returns: stroke color of shape or symbol used to mark data value plot point(s).

8.
```
plot_point_style & stroke_color(const svg_color & f) ;
```

Set stroke color of shape or symbol used to mark data value plot point(s).

Returns: plot_point_style& to make chainable.

9.
```
text_style & style() const;
```

Returns: text_style& To allow control of symbol font, size, decoration etc.

10.
```
plot_point_style & style(text_style ts) ;
```

Returns: plot_point_style& to make chainable.

11.
```
std::string & symbols() ;
```

Returns: plot point marking symbol (only effective if .shape(symbol) used).

12.
```
plot_point_style & symbols(const std::string s) ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Override default symbol "X" - only effective if .shape(symbol) used.

Returns: plot_point_style& to make chainable.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Class svg_style

boost::svg::svg_style — basic SVG stroke, fill colors and width, and their switches.

Synopsis

```
// In header: <boost/svg_plot/svg_style.hpp>


class svg_style {
public:
  // construct/copy/destruct
  svg_style(const svg_color &, const svg_color &, unsigned int);
  svg_style();

  // public member functions
  svg_color fill_color() const;
  svg_style & fill_color(const svg_color &) ;
  bool fill_on() const;
  svg_style & fill_on(bool) ;
  bool operator!=(svg_style &) ;
  bool operator==(svg_style &) ;
  svg_color stroke_color() const;
  svg_style & stroke_color(const svg_color &) ;
  bool stroke_on() const;
  svg_style & stroke_on(bool) ;
  double stroke_width() const;
  svg_style & stroke_width(double) ;
  bool width_on() const;
  svg_style & width_on(bool) ;
  void write(std::ostream &) ;
  svg_color fill_;  // Color of SVG fill.
  bool fill_on_;  // true if fill to be specified.
  svg_color stroke_;  // Color of SVG stroke (line or outline).
  bool stroke_on_;  // true if stroke is to be specified.
  double width_;  // Width of line. Only valid if > 0 & width_on_ == true.
  bool width_on_;  // true if width to be specified.
};
```

Description

This is the style information for any group (g) tag. This could be expanded to include more data from the SVG standard.

There are some strange effects for text on some browsers (Firefox especially) when only stroke is specified. fill is interpreted as black, and the font outline is fuzzy and bolder. <g id="title" stroke="rgb(255,0,0)"> .. is red border and black fill. (because created as a graphic not a builtin font?) <g id="title" fill="rgb(255,0,0)"> .. is red sharp font. <g id="title" stroke="rgb(255,0,0)" fill="rgb(255,0,0)"> red and red fill also fuzzy. So for text, only specific the fill unless a different outline is really wanted. Defaults for text provide a built-in glyph, for example for title: <g id="title"> <text x="250" y="36" text-anchor="middle" font-size="18" font-family="Verdana"> Plot of data </text> </g> and this is not a graphic.

`svg_style` public construct/copy/destruct

1. ```
   svg_style(const svg_color & stroke, const svg_color & fill,
             unsigned int width);
   ```

   Construct svg_style with specified fill and stroke colors, and width.

2. ```
   svg_style();
   ```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

`svg_style` public member functions

1. ```
   svg_color fill_color() const;
   ```

   Returns: SVG fill color.

2. ```
   svg_style & fill_color(const svg_color & col) ;
   ```

   Set fill color (and set fill on true, unless color is blank).

   Returns: svg_style& to make chainable.

3. ```
   bool fill_on() const;
   ```

   Returns: true if fill wanted.

4. ```
   svg_style & fill_on(bool is) ;
   ```

   Set fill is wanted.

   Returns: svg_style& to make chainable.

5. ```
   bool operator!=(svg_style & s) ;
   ```

   Compare svg_styles (for inequality).

6. ```
   bool operator==(svg_style & s) ;
   ```

   Compare svg_styles.

7. ```
   svg_color stroke_color() const;
   ```

   Returns: SVG stroke color.

8. ```
   svg_style & stroke_color(const svg_color & col) ;
   ```

   Set stroke color (and set stroke on).

   Returns: svg_style& to make chainable.

9. ```
   bool stroke_on() const;
   ```

   Returns: true if SVG stroke is on.

10. ```
    svg_style & stroke_on(bool is) ;
    ```

    Set true if SVG stroke is wanted.

    Returns: svg_style& to make chainable.

11. ```
    double stroke_width() const;
    ```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/
    ```

Returns: SVG stroke width.

12.
```
svg_style & stroke_width(double width) ;
```

Set stroke width (and set width on).

Returns: svg_style& to make chainable.

13.
```
bool width_on() const;
```

Returns: true if to use SVG stroke width.

14.
```
svg_style & width_on(bool is) ;
```

Set true to use SVG stroke width.

Returns: svg_style& to make chainable.

15.
```
void write(std::ostream & os) ;
```

Write any stroke, fill colors and/or width info to SVG XML document.

Example output: <g id="yMinorTicks" stroke="rgb(0,0,0)" stroke-width="1">

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Class text_style

boost::svg::text_style — font family, font size, weight, style, stretch & decoration.

Synopsis

```
// In header: <boost/svg_plot/svg_style.hpp>


class text_style {
public:
  // construct/copy/destruct
  text_style(int = 12, const std::string & = "", const std::string & = "",
             const std::string & = "", const std::string & = "",
             const std::string & = "");

  // public member functions
  const std::string & font_decoration() const;
  text_style & font_decoration(const std::string &) ;
  const std::string & font_family() const;
  text_style & font_family(const std::string &) ;
  int font_size() const;
  text_style & font_size(unsigned int) ;
  const std::string & font_stretch() const;
  text_style & font_stretch(const std::string &) ;
  const std::string & font_style() const;
  text_style & font_style(const std::string &) ;
  const std::string & font_weight() const;
  text_style & font_weight(const std::string &) ;
  bool operator!=(const text_style &) ;
  bool operator==(const text_style &) ;
  std::string decoration_;  // Font decoration, examples: "underline" | "overline" | "line-
through".
  std::string font_family_;  // Font family, examples: "Arial", "Times New Roman", "Verdana", ↵
"Lucida Sans Unicode".
  int font_size_;  // Font size (SVG units, default pixels).
  std::string stretch_;  // Font stretch, examples: normal | wider | narrower.
  std::string style_;  // Font weight, examples: normal | bold | italic | oblique.
  std::string weight_;  // Font style, examples: "bold", "normal".
};
```

Description

`text_style` public construct/copy/destruct

1.
```
text_style(int size = 12, const std::string & font = "",
           const std::string & style = "", const std::string & weight = "",
           const std::string & stretch = "",
           const std::string & decoration = "");
```

`text_style` public member functions

1.
```
const std::string & font_decoration() const;
```

Returns: font decoration.

2.
```
text_style & font_decoration(const std::string & s) ;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Set font decoration. Examples: "underline" | "overline" | "line-through" http://www.croczilla.com/~alex/conformance_suite/svg/text-deco-01-b.svg tests line-through and underline. But implementation varies.

Returns: reference to text_style to make chainable.

3.
```
const std::string & font_family() const;
```

Returns: font family as string.

4.
```
text_style & font_family(const std::string & s) ;
```

Set font family, for example: "Arial", "Times New Roman", "Verdana", "Lucida Sans Unicode".

Default for browser is sans with Firefox & IE but serif with Opera. See also browser conformance test at http://www.croczilla.com/~alex/conformance_suite/svg/text-fonts-01-t.svg which tests three styles of font, serif, sans-serif and mono-spaced. <text font-family="Georgia, 'Minion Web', 'Times New Roman', Times, 'MS PMincho', Heisei-Mincho, serif " x="20" y="80">A serifed face</text>\n <text font-family="Arial, 'Arial Unicode', 'Myriad Web', Geneva, 'Lucida Sans Unicode', 'MS PGothic', Osaka, sans-serif " x="20" y="160">A sans-serif face</text>\n <text font-family="'Lucida Console', 'Courier New', Courier, Monaco, 'MS Gothic', Osaka-Mono, monospace" x="20" y="240">A mono (iW) face</text>

Returns: reference to text_style to make chainable.

5.
```
int font_size() const;
```

Returns: font size (svg units, usually pixels).

6.
```
text_style & font_size(unsigned int i) ;
```

Set font size (svg units usually pixels) default 10.

Returns: text_style& to make chainable.

reference to text_style to make chainable.

7.
```
const std::string & font_stretch() const;
```

Returns: font stretch, for example: normal | wider | narrower .

8.
```
text_style & font_stretch(const std::string & s) ;
```

Examples: "wider" but implementation by browsers varies. font-stretch: normal | wider | narrower ...

Returns: reference to text_style to make chainable.

9.
```
const std::string & font_style() const;
```

font-style: normal | bold | italic | oblique. Example "normal" is default.
Returns: font style.

10.
```
text_style & font_style(const std::string & s) ;
```

Set font style. Examples: my_text_style.font_style("italic"); See also browser conformance tests: http://www.croczilla.com/~alex/conformance_suite/svg/text-fonts-02-t.svg

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Returns: reference to text_style to make chainable.

11.
```
const std::string & font_weight() const;
```

12.
```
text_style & font_weight(const std::string & s) ;
```

svg font-weight: normal | bold | bolder | lighter | 100 | 200 .. 900 Examples: "bold", "normal" http://www.croczilla.com/~alex/conformance_suite/svg/text-fonts-02-t.svg tests conformance. Only two weights, "bold", "normal", are supported by Firefox, Opera, Inkscape.

Returns: reference to text_style to make chainable.

13.
```
bool operator!=(const text_style & ts) ;
```

Compare text_style for inequality (needed for testing).

14.
```
bool operator==(const text_style & ts) ;
```

Compare text_style for equality (needed for testing).

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Class ticks_labels_style

boost::svg::ticks_labels_style — Style of the X and Y axes ticks, grids and their tick value labels.

Synopsis

```
// In header: <boost/svg_plot/svg_style.hpp>


class ticks_labels_style {
public:
  // construct/copy/destruct
  ticks_labels_style(dim = X, const text_style & = no_style, double = 10.,
                     double = -10., double = 2., unsigned int = 4);

  // public member functions
  double label_length(double) ;
  double longest_label() ;
  ticks_labels_style & major_value_labels_side(int) ;
  int major_value_labels_side() const;
  ticks_labels_style & use_down_ticks(bool) ;
  bool use_down_ticks() const;
  ticks_labels_style & use_up_ticks(bool) ;
  bool use_up_ticks() const;
  dim dim_;  // X, Y, or None.
  bool down_ticks_on_;  // Draw ticks down from horizontal X-axis line.
  double label_max_length_;  // width (in SVG units, pixels) of longest value label text on axis.
  double label_max_space_;  // Space (SVG units, pixels) needed for value label adjusted for ro↵
tation.
  rotate_style label_rotation_;  // Direction axis value labels written.
  bool left_ticks_on_;  // Draw ticks left from vertical Y-axis line.
  svg_color major_grid_color_;  // Color of major grid lines.
  bool major_grid_on_;  // Draw X grid at major ticks.
  double major_grid_width_;  // Width of major grid lines.
  double major_interval_;  // Stride or interval between major x ticks (Cartesian units).
  svg_color major_tick_color_;  // Color (stroke) of tick lines.
  double major_tick_length_;  // Length of major tick lines.
  double major_tick_width_;  // Width of major tick lines.
  int major_value_labels_side_;  // Which side of axis for label values for major ticks.
  double max_;  // Maximum x value (Cartesian units).
  double min_;  // Minimum x value (Cartesian units).
  svg_color minor_grid_color_;  // color of minor grid lines.
  bool minor_grid_on_;  // Draw X grid at minor ticks.
  double minor_grid_width_;  // Wdith of minor grid lines.
  double minor_interval_;  // Interval (Cartesian units) between minor ticks.
  svg_color minor_tick_color_;  // Color (stroke) of tick lines.
  double minor_tick_length_;  // Length of minor tick lines.
  double minor_tick_width_;  // Width of minor tick lines.
  unsigned int num_minor_ticks_;  // number of minor ticks, eg 4 gives major 0, minor 1,2,3,4, ↵
major 5 (All units in svg units, default pixels).
  bool right_ticks_on_;  // Draw ticks right from vertical Y-axis line.
  bool strip_e0s_;  // If redundant zero, + and e are to be stripped, for example "+1.000e3" to ↵
"1e3".
  int ticks_on_window_or_on_axis_;  // Value labels & ticks on a plot window border (rather than ↵
on X or Y-axis). For Y-axis -1 = left, 0 = false = on X-axis, +1 = right. Default -1 to left of ↵
plot window. For X-axis -1 = bottom, 0 = false = on Y-axis, +1 = top. Default -1 below bottom ↵
of plot window. 0 = false puts the ticks and their labels on the X or Y axis line which may be ↵
in the middle of the plot. For 1D the default overrides the constructor default of -1 below, to ↵
tick and value label the X-axis. For 2D the default is left at -1, to use bottom and left of ↵
plot window to ticka nd value label X and Y-axis.
  bool up_ticks_on_;  // Draw ticks up from horizontal X-axis line.
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
  std::ios_base::fmtflags value_ioflags_;  // IO formatting flags for the axis default ↵
std::ios::dec.
  text_style value_label_style_;  // text style (font, size...) for value labels.
  int value_precision_;  // Precision for tick value labels, usually 3 will suffice.
  svg_color values_color_;  // Color of tick values labels.
};
```

Description

But NOT the X and Y axes lines. These can be either on the axis lines or on the plot window edge(s), (because different styles for x and y are possible).

`ticks_labels_style` public construct/copy/destruct

1.
```
ticks_labels_style(dim d = X, const text_style & style = no_style,
                   double max = 10., double min = -10.,
                   double major_interval = 2.,
                   unsigned int num_minor_ticks = 4);
```

`ticks_labels_style` public member functions

1.
```
double label_length(double value) ;
```

Find the length of label (like "1.23E-5") for a value.

2.
```
double longest_label() ;
```

Update label_max_length_ with the longest value label as pixels, return the count of digits etc.

3.
```
ticks_labels_style & major_value_labels_side(int is) ;
```

Set side for tick value labels: left (<0), none (==0) or right (>0).

Returns: ticks_labels_style& to make chainable.

4.
```
int major_value_labels_side() const;
```

Returns: side for tick value labels: left (<0), none (==0) or right (>0).

5.
```
ticks_labels_style & use_down_ticks(bool side) ;
```

Set true if to draw ticks down from horizontal X-axis line.

Returns: ticks_labels_style& to make chainable.

6.
```
bool use_down_ticks() const;
```

Returns: true if to draw ticks down from horizontal X-axis line.

7.
```
ticks_labels_style & use_up_ticks(bool is) ;
```

Set true to draw ticks up from horizontal X-axis line.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Returns: ticks_labels_style& to make chainable.

8.
```
bool use_up_ticks() const;
```

Returns: true if to draw ticks up from horizontal X-axis line.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Class value_style

boost::svg::value_style — Data series point value label information, text, color, orientation, (uncertainty & df).

Synopsis

```
// In header: <boost/svg_plot/svg_style.hpp>


class value_style {
public:
  // construct/copy/destruct
  value_style(rotate_style, int, std::ios_base::fmtflags, bool, text_style,
              const svg_color &, svg_color, bool, bool, std::string,
              std::string, std::string);
  value_style();
  svg_color df_color_;  // Color for degrees for freedom, for example: 99 in "1.23 +-0.02 (99)".
  bool df_on_;  // If a degrees of freedom estimate is to be appended.
  svg_color fill_color_;  // Fill color for value.
  svg_color plusminus_color_;  // Color for uncertainty, for example: 0.02 in "1.23 +-0.02".
  bool plusminus_on_;  // If an uncertainty estimate is to be appended (as + or - value).
  std::string prefix_;  // Prefix to data point value, default none, but typically "[".
  std::string separator_;  // Separator between x and y values, if both on same line (none if ↵
only X or only Y, or Y below X).
  bool strip_e0s_;  // If true, then unnecessary zeros and + sign will be stripped to reduce ↵
length.
  svg_color stroke_color_;  // Stroke color for value.
  std::string suffix_;  // Suffix to data point value, default none, but typically "]".
  std::ios_base::fmtflags value_ioflags_;  // Control of scientific, fixed, hex etc.
  rotate_style value_label_rotation_;
  int value_precision_;  // Decimal digits of precision of value.
  text_style values_text_style_;  // Font etc used for data point value marking.
};
```

Description

For example, to output: 5.123 +- 0.01 (19). Uncertainty and degrees of freedom estimate. Prefix, separator and suffix allow X and Y values to be together on one line, for example [1.23+- 0.01 (3), 4.56 +-0.2 (10)] Used in draw_plot_point_values (note plural - not used in draw_plot_point_value) where X value_style is used to provide the prefix and separator, and Y value_style to provide the suffix. Prefix, separator and suffix are ignored when X or Y are shown separately using draw_plot_point_value.

`value_style` public construct/copy/destruct

1.
```
value_style(rotate_style r, int p, std::ios_base::fmtflags f, bool s,
            text_style ts, const svg_color & scol, svg_color fcol, bool pm,
            bool df, std::string pre, std::string sep, std::string suf);
```

2.
```
value_style();
```

Default style for a data point value label.

< Constructor Data point value label style (provides default color and font).

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Global no_style

boost::svg::no_style — Text style that uses all constructor defaults.

Synopsis

```
// In header: <boost/svg_plot/svg_style.hpp>

text_style no_style;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Global wh

boost::svg::wh — font text width/height ratio.

Synopsis

```
// In header: <boost/svg_plot/svg_style.hpp>

static const double wh;
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Function operator!=

boost::svg::operator!=

Synopsis

```
// In header: <boost/svg_plot/svg_style.hpp>


bool operator!=(const text_style & lhs, const text_style & rhs);
```

Description

Compare two text_style for equality. Note operator== and operator << both needed to use Boost.Test. (But can be avoided with a macro define).

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Function operator<<

boost::svg::operator<<

Synopsis

```
// In header: <boost/svg_plot/svg_style.hpp>


std::ostream & operator<<(std::ostream & os, plot_line_style p);
```

Description

Output description of plot_line_style. (mainly useful for diagnosis).

Example Usage: plot_line_style p; cout << p << endl; Outputs: point_line_style(RGB(0,0,0), blank, line, no bezier)

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Function operator<<

boost::svg::operator<<

Synopsis

```
// In header: <boost/svg_plot/svg_style.hpp>


std::ostream & operator<<(std::ostream & os, plot_point_style p);
```

Description

Output description of data value plot point marker(s).

Example: plot_point_style p; cout << p << endl; Outputs: point_style(1, RGB(0,0,0), RGB(0,0,0), 10, X)

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Function operator<<

boost::svg::operator<<

Synopsis

```
// In header: <boost/svg_plot/svg_style.hpp>


std::ostream & operator<<(std::ostream & os, const text_style & ts);
```

Description

Output a text style as a text string (mainly useful for diagnostic use).

Usage: text_style ts(12, "Arial", "italic", "bold", "", ""); cout << t << endl; Outputs: text_style(18, "Arial", "italic", "bold", "", "")

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Function operator<<

boost::svg::operator<<

Synopsis

```
// In header: <boost/svg_plot/svg_style.hpp>


std::ostream & operator<<(std::ostream & os, const svg_style & s);
```

Description

Output a string description of a svg_style. Usage: svg_style my_svg_style; cout << my_svg_style << endl; Outputs: svg_style(RGB(0,0,0), RGB(0,0,0), 0, no fill, no stroke, no width)

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Function operator==

boost::svg::operator==

Synopsis

```
// In header: <boost/svg_plot/svg_style.hpp>


bool operator==(const text_style & lhs, const text_style & rhs);
```

Description

Compare two text_style for equality Note operator== and operator << both needed to use Boost.text. (But can be avoided with a macro define).

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Function string_svg_length

boost::svg::string_svg_length

Synopsis

```
// In header: <boost/svg_plot/svg_style.hpp>


double string_svg_length(const std::string & s, const text_style & style);
```

Description

http://www.w3.org/TR/SVG/text.html#FontSizeProperty Font size is the height of the text's font, so width = wh * font_size.

Even after reading http://www.w3.org/TR/SVG/fonts.html, unclear how to determine the exact width of digits, so an arbitrary average width height ratio wh = 0.7 is used as a good approximation.

Header <boost/svg_plot/uncertain.hpp>

Class for storing Uncertainties and handling simple Propagation of according to a pure Gaussian model.

This simplifed version assuming uncorrelated uncertainties (the common case) is based on code by Evan Manning (manning@alumni.caltech.edu) Evan Marshal Manning, C/C++ Users Journal, March 1996 page 29 to 38. original downloaded from ftp://beowulf.jpl.nasa.gov/pub/manning This is a simple model of uncertainties, designed to accompany an article published in C/C++ Users Journal March 1996. A fuller collection of even fancier classes also given in UReal.h. And also based on a extended version including uncertainty as standard deviation & its uncertainty as degrees of freedom, and other information about the value added Paul A Bristow from 31 Mar 98.

http://en.wikipedia.org/wiki/Plus-minus_sign

Paul A. Bristow

Mar 2009

```
namespace boost {
  namespace svg {
    class unc;

    static const double plusminus;    // Nominal factor of 2 (strictly 1.96) corresponds to 95% ↵
confidence limit.
    std::ostream & operator<<(std::ostream &, const std::pair< unc, unc > &);
    std::ostream & operator<<(std::ostream &, const unc &);
    template<> float unc_of(unc);
    template<class T > float unc_of(T);
    template<class T > std::pair< float, float > uncs_of(std::pair< T, T >);
    template<class T > std::pair< float, float > uncs_of(T);
    template<> double value_of(unc);
    template<class T > double value_of(T);
    template<class T >
      std::pair< double, double > values_of(std::pair< const unc, unc >);
    template<class T >
      std::pair< double, double > values_of(std::pair< const T, T >);
    template<class T > std::pair< double, double > values_of(T);
  }
}
```


XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Class unc

boost::svg::unc — Uncertain class for storing an observed or measured value together with information about its uncertainty (previously called 'error' or 'plusminus', but now deprecated) represented nominally one standard deviation (but displayed as a multiple, usually two standard deviations).

Synopsis

```
// In header: <boost/svg_plot/uncertain.hpp>


class unc {
public:
  // construct/copy/destruct
  unc(double, float, short unsigned, short unsigned);
  unc();
  unc& operator=(const unc &);

  // public member functions
  void deg_free(short) ;
  short unsigned deg_free() const;
  bool operator<(unc &) const;
  bool operator<(const unc &) const;
  bool operator==(const unc &) const;
  void types(short) ;
  short unsigned types() const;
  void uncertainty(float) ;
  float uncertainty() const;
  void value(double) ;
  double value() const;
  short unsigned deg_free_;
  short unsigned types_;  // Reserved for other information about the value.
  float uncertainty_;  // Estimate of uncertainty, typically one standard deviation. Negative ↵
values mean that uncertainty is not defined.
  double value_;  // Most likely value, typically the mean.
};
```

Description

This version assumes uncorrelated uncertainties (by far the most common case).

http://www.measurementuncertainty.org/ International Vocabulary of Basic and General Terms in Metrology; ISO/TAG 4 1994 ISO, Guide to the expression of uncertainty in measurement, ISO, Geneva, 1993. Eurochem, Quantifying uncertainty in analytical measurements.

unc public construct/copy/destruct

1. ```
   unc(double v, float u, short unsigned df, short unsigned ty);
   ```

2. ```
   unc();
   ```

3. ```
   unc& operator=(const unc & rhs);
   ```

   Assignment simply copies all values, including those with 'undefined' status.

   to make chainable.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/
   ```

unc public member functions

1. ```
   void deg_free(short unsigned) ;
   ```

   Set degrees of freedom, usually = number of observations -1;

2. ```
   short unsigned deg_free() const;
   ```

   Returns: Degrees of freedom, usually the number of observations -1.

3. ```
   bool operator<(unc & rhs) const;
   ```

   Less operator only compares the value, ignoring any uncertainty information.

4. ```
   bool operator<(const unc & rhs) const;
   ```

   Less operator only compares the value, ignoring any uncertainty information.

5. ```
   bool operator==(const unc & rhs) const;
   ```

   Equality operator only compares the value, ignoring any uncertainty information.

6. ```
   void types(short unsigned) ;
   ```

   Set other information about the uncertain value.

7. ```
   short unsigned types() const;
   ```

   Returns: Other information about the uncertain value.

8. ```
   void uncertainty(float) ;
   ```

   Set estimate of uncertainty, typically standard deviation.

9. ```
   float uncertainty() const;
   ```

   Returns: Estimate of uncertainty, typically standard deviation.

10. ```
    void value(double) ;
    ```

    Set most likely value, typically the mean.

11. ```
    double value() const;
    ```

    Returns: Most likely value, typically the mean.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/
    ```

Global plusminus

boost::svg::plusminus — Nominal factor of 2 (strictly 1.96) corresponds to 95% confidence limit.

Synopsis

```
// In header: <boost/svg_plot/uncertain.hpp>

static const double plusminus;
```

Description

Number of standard deviations used for plusminus text display.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Function operator<<

boost::svg::operator<<

Synopsis

```
// In header: <boost/svg_plot/uncertain.hpp>


std::ostream & operator<<(std::ostream & os, const std::pair< unc, unc > & u);
```

Description

Output a pair (X and Y) value with (if defined) uncertainty and degrees of freedom. For example: "1.23 +/- 0.01 (13), 3.45 +/- 0.06 (78)".

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Function operator<<

boost::svg::operator<<

Synopsis

```
// In header: <boost/svg_plot/uncertain.hpp>


std::ostream & operator<<(std::ostream & os, const unc & u);
```

Description

Output an value with (if defined) uncertainty and degrees of freedom (and type). For example: "1.23 +/- 0.01 (13)". /details Note that the uncertainty is input and stored as one standard deviation, but output multiplied for a user configurable 'confidence factor' plusminus, default two for about 95% confidence (but could also be one for 67% or 3 for 99% confidence).

Note that the plus or minus can be output using several methods.

256 character 8-bit codepage plusminus symbol octal 361, or

decimal 241 or

hexadecimal F1, or

Unicode space plusminus glyph, or os << " +or-" << u.uncertainty_; Plain ANSI 7 bit code chars.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Function template unc_of

boost::svg::unc_of

Synopsis

```
// In header: <boost/svg_plot/uncertain.hpp>


template<> float unc_of(unc v);
```

Description

Returns: unc.uncertainty() as a float.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Function template unc_of

boost::svg::unc_of

Synopsis

```
// In header: <boost/svg_plot/uncertain.hpp>


template<class T > float unc_of(T);
```

Description

Allow uncertainty (standard deviation) part of variables of class unc to be assigned to, and compared with float.

Returns: zero always (because no uncertainty information is available for built-in double, float, or long double).

zero always (because no uncertainty information is available for built-in double, float, or long double).

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Function template uncs_of

boost::svg::uncs_of — Get uncertainties (standard deviation) of a pair of values.

Synopsis

```
// In header: <boost/svg_plot/uncertain.hpp>


template<class T > std::pair< float, float > uncs_of(std::pair< T, T > vp);
```

Description

Returns: uncertainty parts (if any) as a pair of floats.

uncertainty parts (if any) as a pair of floats.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Function template uncs_of

boost::svg::uncs_of — Get uncertainties (standard deviation) of a pair of values.

Synopsis

```
// In header: <boost/svg_plot/uncertain.hpp>


template<class T > std::pair< float, float > uncs_of(T);
```

Description

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Function template value_of

boost::svg::value_of

Synopsis

```
// In header: <boost/svg_plot/uncertain.hpp>


template<> double value_of(unc v);
```

Description

Returns: unc.value() as a double.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Function template value_of

boost::svg::value_of

Synopsis

```
// In header: <boost/svg_plot/uncertain.hpp>


template<class T > double value_of(T v);
```

Description

Allow value part of variables of class unc to be assigned to, and compared with double.

Returns: value as a double.

value as a double.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Function template values_of

boost::svg::values_of

Synopsis

```
// In header: <boost/svg_plot/uncertain.hpp>


template<class T >
  std::pair< double, double > values_of(std::pair< const unc, unc > up);
```

Description

Returns: value (part) as a pair of doubles.

value (part) as a pair of doubles.

value (part) as a pair of doubles.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Function template values_of

boost::svg::values_of

Synopsis

```
// In header: <boost/svg_plot/uncertain.hpp>


template<class T >
  std::pair< double, double > values_of(std::pair< const T, T > vp);
```

Description

<

Returns: values of a pair of double values.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Function template values_of

boost::svg::values_of

Synopsis

```
// In header: <boost/svg_plot/uncertain.hpp>


template<class T > std::pair< double, double > values_of(T);
```

Description

Implementation & Rationale

Switches for axis labels and autoscaling

If a axis label string is provided, or autoscaling is requested, it is assumed that it should be displayed/acted on. So there functions have the effect of switching these options on, as if , for example:

`x_autoscale(true); y_label_on(true);`

This avoids users providing an axis label string and then wondering why it appears to have no effect.

It also avoids wasting plot space for empty labels.

It is still possible to switch these options off, for example with:

`x_autoscale(false); y_label_on(false);`

Number of Minor ticks

`x_num_minor_ticks()` and `x_num_minor_ticks()`

are provided control the number of minor ticks between the major ticks.

There are a total of x_num_minor_ticks + 1 ticks for each major tick, for example, a major on 0, minor on 1,2,3,4, major on 5 ...

For integer usually binary, octal and hexadecimal, num_minor_ticks = 2[super]n -1 are useful, for example 1 if major ticks are even will give minor ticks on odd values), 3 if major on 0, minor on 1, 2, 3 and major on 4... 7 if major on 0, minor on 1,2,3,4,5,6,7 and major on 8 ... 15 if major on 0, minor on 1,2,3,4,5,6,7, 8,9,A,B,C,D,E,F, major on 0x10...

```
For decimal based values, num_minor_ticks 1, 4, 9 are useful, for example:
1 if major are even, major 0, minor 1, major 2 ...
4 if major on 0, minor on 1,2,3,4, major on 5 ...
9 if major on 0, minor on 1,2,3,4,5,6,7,8,9, major on 10
```

This section provide more information about this implementation and some of the rationale for desing decisions.

SVG Specification

SVG version 1.1 was used in the design of this version but SVG 1.2 draft specification is also available and appears to be final and designed to be backward compatible. No changes that affect the code produced have been detected from a quick perusal of this document. It is probable but untested that the svg files produced will also comply with the Tiny SVG for mobiles specification.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Design

SVG would be a flat format if it weren't for the <g> elements: these make it parse into a tree.

If each element has its own style (fill, stroke etc) then the .svg file size would be increased. (This style is used by other packages that output svg and often leads to larger file sizes).

So the group element is used with each type given an id to allow reference back to it.

svg_style_details.hpp contains a list of these groups, for example: PLOT_X_AXIS, PLOT_TITLE... indexing an array of string document id "yAxis", "title"...

and these can be seen, with style information in the output, for example:

```
<g id="yAxis" stroke="rgb(0,0,0)"></g>

<g id="title">
<text x="250" y="20" text-anchor=" middle" font-size="20">Demo 1D plot</text></g>
```

In the general case, the most that occurs grouped together is the style information: axis lines all share the same style information, points in a series all share the same color information, and this is also a logical grouping. One can can add a series, and then come back later and change the <g> element above the points, which is a single change, and have the change reflected on every point in the series. It seems this is the most logical way to represent the data in memory.

Economising on .svg File Size

Some of the factors affecting the file size are:

1. Excessive precision of data points.

If plots are to be viewed only at a modest size, then the precision of x and y coordinates does not need to be higher than about 1 in 1000, so a precision of about 3 decimal digits will suffice.

A default of 3 decimal digits has been chosen, but the precision can be controlled to permit a higher resolution, for example on a map that is printed at 2000 dpi. or even only 2 decimal digits for a display on a mobile, where file transfer speed may make reducing file size important.

1. Excessive data points. When plotting functions, it is sensible to avoid using more data points than are justified by the resolution. Again 1 in 1000 is a typical number, and by using bezier curve fitting, as few as 100 points should be sufficient to produce a visibly perfectly smooth curve for 'well-behaved' functions.

Plotting extremely large datasets from files may cause memory overflow, as with any STL container held in memory. It may be necessary to perform some averaging or smoothing, or just using a 'stride' to select, for example, every 100th value to be plotted.

1. Redundant style specifications.

The use of the g or group element has been used to try to reduce repeated (and thus redundant) style specifications. More efficient use of groups may be possible.

1. Not using default style and attributes where possible.

Some effort has been made to use defaults, but more may be possible.

Adding document information to the svg Image files

Several ways of adding useful information to the svg document are provided. If the strings are null (the default) nothing is output.

Member function svg& description(const std::string) allows output like:

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
<desc>My Document description</desc>

svg& document_title(const std::string)

<title>My Document title</title>
```

(Note that this is **not** the same as the title of the plot.)

Adding Copyright and License conditions for the SVG Image files

In general, setting values for items like copyright_holder will ensure that they appear in the SVG document both as XML and as XML comment. If none are set, nothing will be output, to minimize file size.

www.w3.org/TR/2004/WD-SVG12-20041027 discusses primary documents, of type .svg in section 17.3 Adding Copyright information to an SVG document: SVG encourages the use of a common metadata format for inclusion of copyright information. Metadata relevant to the data copyright of the entire document should be added to metadata element of the topmost svg element. This allows the author to unambiguously state the licensing terms for the entire document. The scheme may also be used elsewhere in the document, for pieces that have different licensing. For example, an SVG font may have specific licensing details expressed in its own metadata element.

Note that inclusion of this metadata does not provide the author with a method in which to protect or enforce their copyright, it simply bundles the copyright information with the content in a defined manner. Providing methods, technical or non-technical, for data protection is currently beyond the scope of the SVG specification.

This does not exclude the use of other metadata schemes.

A simple way of adding metadata for copyright is provided by functions to set

```
copyright_holder(std::string);
copyright_date(std::string);
```

(as well as description and document_title, and .svg filename if any).

For example, setting these will incorporate an XML comment

```
<!-- SVG Plot Copyright Paul A. Bristow 2007 -->
```

and also as meta data:

```
<meta name="copyright" content="Paul A. Bristow" />
<meta name="date" content="2007" />
```

The Creative Commons Metadata Set is also provided as an option in this implementation, following this example.

The Creative Commons License is one method of providing license terms in a machine-readable format.

Typical data added to the file would be XML like this:

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
<metadata>
  <rdf:RDF xmlns:cc="http://web.resource.org/cc/"
           xmlns:dc="http://purl.org/dc/elements/1.1/"
           xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <cc:work rdf:about="filename.svg">
      <!-- insert filename or title here -->
      <dc:title>Plot title</dc:title>
      <dc:creator>Boost.Plot</dc:creator>
      <dc:author>Paul A. Bristow</dc:author>
      <dc:format>application/xhtml+xml+svg</dc:format>
        <cc:license rdf:about="http://creativecommons.org/licenses/by-sa/.0/">
           <cc:requires rdf:resource="http://web.resource.org/cc/Notice"/>
           <cc:permits rdf:resource="http://web.resource.org/cc/Reproduction"/>
           <cc:permits rdf:resource="http://web.resource.org/cc/Distribution"/>
           <cc:requires rdf:resource="http://web.resource.org/cc/Attribution"/>
           <cc:permits rdf:resource="http://web.resource.org/cc/CommercialUse"/>
        </cc:license>
    </cc:work>
  </rdf:RDF>
</metadata>
```

cc:permits can also be cc:requires or cc:prohibits, as required.

A license that permits reproduction, distribution and commercial use, but requires both notice & attribution is probably most suitable for Boost documents as is therefore chosen as the default.

This license can be included by calling svg member function `is_license(true)`. If this license will be included can be discovered by calling svg member function `is_license()`.

Similarly functions const std::string license_reproduction(); const std::string license_distribution(); const std::string license_attribution(); const std::string license_commercialuse() allow you to find the current license requirements.

RDF is the metadata format chosen by Creative Commons.

and references the resource (cc) at http://web.resource.org/cc/ using the definition of its XML namespace:

```
rdf:RDF xmlns:cc="http://web.resource.org/cc/"
```

Options are summarized.

A way of seting the author is also provided, in case the copyright has been assigned to someone else, for example a publisher. svg set and get member functions:

```
author(const std::string);
const std::string author();
```

Using Unicode Symbols (usually Math Sumbols and Greek letters)

Unicode symbols that work on most browsers in html are listed at:

html4_symbols, reference/html40/entities/symbols, and demos.

However support for Unicode in SVG is much less fully implemented and displayed results are variable.

The Unicode value in decimal 9830 or hex x2666 must be prefixed with `&` and terminated with `;` for example, `&x2666`; for xml and then enveloped with "" to convert to a const std::string, for example: "♦" for diamond.

Thus diamond can be used a point marker.

Similarly for greek in title, legend and axes:

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
.legend_title("My Legend &#956;") // generates <em>&#956;</em>  greek mu
```

Subscript and superscript in title, legend and labels

It is very common need to show superscript, in units for example, area (m[super 2]), and subscript for a[sub 0, a[sub 2]...

Sadly, although these needs are both OK with html, showing sub and superscripts in svg doesn't work well as yet because browsers don't handle <sub> or <sup>, nor baseline-shift.

The only reasonably widely supported feature is Unicode symbols for superscript 1, 2 and 3 (only). For example, Latin1_symbols: sup1 &185; sup2 &178; sup3 &179; work on Firefox & IE6/7.

Butࠗ ࠛ ࠪ - show just square boxes.

The SVG specification covers <sub, <super>, and the general baseline-shift, but these are just not implemented yet.

2.8 Superscripts and Subscripts symbols

At Nov 2007, the following commands don't have very useful coverage. <tspan baseline-shift = "50%" font-stretch = "wider" font-variant = "small-caps" > 3 </tspan> baseline-shift = "50%" & super and subscript down't work on Firefox 2. but DO on IE6 (and probably 7) font-stretch = "wider" No effect Firefox or IE6. font-variant = "small-caps" No effect Firefox or IE6. <tspan baseline-shift = "50%" >3/4 </tspan>

This has been reported as a bug to Mozilla, and is regarded a duplicate of other reports on sub and super https://bugzilla.mozilla.org/show_bug.cgi?id=401102

but baseline-shift has no effect on Firefox 2. See http://www.imagemagick.org/discourse-server/viewtopic.php?f=3&t=7410&view=previous

The use of <tspan to shift characters **is** feasible, as shown in this example, which displays correctly in all browsers,

and it is also possible to use

```
<svg:g transform="translate(5.062500, -5.367188)">
```

to shift the next letter up, but these parameters are font-size related,* and the svg most verbose.

None of these method provides a **convenient** method of creating the right string for titles, legends, or axis labels, but it may be possible to devise code that does.

Coding style

In general Boost coding style guide and Boost coding guidelines has been used.

Some data members end with _ to avoid name clashes, another widely used convention.

To Do List

This project is still in development: here is a 'known wishes' list. Suggestions to pbristow at hetp.u-net.com.

- Allow **function pointers** to specify functions as input. A typical use case is to just see what a function like `sin(x)` looks like.
- **Radian coordinate system**.
- Implementation of **SVG's DOM**. This will make the SVG class useful for other things than solely making graphs! The existing SVG features implemented are now documented better, so one could consider using it for other purposes.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

- **Automatic scaling** of the axis has been implemented. Feedback on how 'nicely' it works in practice would be welcome. Some of the uglier features of Microsoft Excel autoscaling have been avoided, but opinions on aesthetics vary widely.
- Allow **2D plots of 1D containers** - a very common need, for example for Time Series. Jake had trouble with having functors that update state interact with `make_transform_iterator`, and omitted the feature.
- Make time and/or data labelled axis possible, with datetime a first class citizen type.
- Allow **other image formats**. There are many inherent difficulties with this part. The solution that was explored is allowing the user to pass a functor that traverses the document tree. Generalizing images to an `image` class is fraught with difficulties, as SVG is a tree-based format, which it does not share with many other formats.
- (Note that Inkscape permits conversion to bitmap format, .png etc, and other tools can accept and convert bitmaps. This is used in the Boost.Math documentation. A Python file called generate.sh in the math toolkit shows how to automate this conversion process for many .svg files in a folder using Inkscape in command line mode. $inkscape -d $dpi -e $(cygpath -a -w $pngfile) $(cygpath -a -w $svgfile))
- **Avoid redrawing the entire plot each time**. This is the easiest way to write the program initially, but it would be more efficient if the program could keep track of what has been changed and what hasn't, so that it may be more efficient if lots of images are being produced. This hasn't been found to be a problem in practice so far.
- Allow the user to provide a function object for generating **custom axis labels**, for example, label axes with names of months (Jan, Feb, Mar...) instead of integer representations of months (1, 2, 3...).
- Logarithmic Axes*. This does not look simple. Meanwhile you can take the log of the data.
- Allow an **external stylesheet** to be loaded to style the graph. External stylesheets will allow a standard and easy way to style the document so that users don't have to come up with their own home-grown solutions. This was sketched out, but seemed much more complicated now that very may more options to control features have been added. (The 'first try' code that Jake produced now longer compiles, so has been commented out).

Acknowledgements

Jake Voytko would like to thank the following people:

- **Google**: For offering Summer of Code, and giving me the opportunity to do something I never would have done otherwise.
- **Joaquín Mª López Muñoz**: My GSoC mentor. His proofreading and advice helped shape the project into what it is today, and prevented small problems from becoming major problems.
- **Paul A. Bristow**: For showing an active interest, offering literally dozens of minor and major features he'd like to see, and for helping with Boost.Build.
- **Matias Capeletto**: For showing an active interest, and offering feature suggestions.
- **Sarah Braun**: For helping me pick colors for examples.
- **Boost Community**: For all of the encouragement, suggestions, disagreements, and patience.

Paul A. Bristow would like to thank

- Jake Voytko for setting up the plot package during his Google Summer of Code period in 2007. I hope he will approve of its current state.
- John Maddock for much assistance with problems, and especially with documentation and development of indexing and pdf production.
- Joel de Guzman and Eric Niebler for producing the Quickbook and Doxygen indexing system.
- Daniel James for enhancements to Doxygen with Docbook.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

- And of course all the bits of Boost that proved invaluable.

Class Index

Symbols

1-D Auto scaling Examples
scale_axis, 29
show_1d_plot_settings, 29
x_autoscale, 29
1-D Data Values Examples
show_1d_plot_settings, 41

A

alignment
Header < boost/svg_plot/detail/svg_tag.hpp >, 159
text_element public member functions, 160
author
Header < boost/svg_plot/svg.hpp >, 190
Header < boost/svg_plot/svg_fwd.hpp >, 337
Implementation & Rationale, 384
svg public member functions, 192
autoscale
Demonstration of using 1D data that includes information about its uncertainty, 74
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 205
svg_2d_plot public member functions, 240
svg_boxplot public member functions, 288
autoscale_check_limits
Demonstration of using 1D data that includes information about its uncertainty, 74
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 205
svg_2d_plot public member functions, 240
svg_boxplot public member functions, 288
autoscale_plusminus
Demonstration of using 1D data that includes information about its uncertainty, 74
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 205
svg_2d_plot public member functions, 241
svg_boxplot public member functions, 288
axes_on
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 205
svg_2d_plot public member functions, 241
svg_boxplot public member functions, 288
axis_color
Header < boost/svg_plot/svg_boxplot.hpp >, 280, 319
svg_boxplot public member functions, 288
svg_boxplot_series public member functions, 320

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

axis_line_style
Header < boost/svg_plot/svg_style.hpp >, 340
axis_line_style public member functions
color, 341
position, 341
width, 341
axis_width
Header < boost/svg_plot/svg_boxplot.hpp >, 280, 319
svg_boxplot public member functions, 289
svg_boxplot_series public member functions, 321

B

background_border_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 205
svg_2d_plot public member functions, 241
svg_boxplot public member functions, 289
background_border_width
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 205
svg_2d_plot public member functions, 241
svg_boxplot public member functions, 289
background_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 205
svg_2d_plot public member functions, 241
svg_boxplot public member functions, 289
bar_opt
bar_style public member functions, 342
Header < boost/svg_plot/svg_2d_plot.hpp >, 275
Header < boost/svg_plot/svg_style.hpp >, 342
svg_2d_plot_series public member functions, 276
bar_style
Header < boost/svg_plot/svg_style.hpp >, 342
bar_style public member functions
bar_opt, 342
width, 343
bar_width
Header < boost/svg_plot/svg_2d_plot.hpp >, 275
svg_2d_plot_series public member functions, 276
bezier_on
Header < boost/svg_plot/svg_1d_plot.hpp >, 226
Header < boost/svg_plot/svg_2d_plot.hpp >, 275
svg_1d_plot_series public member functions, 227
svg_2d_plot_series public member functions, 277
boost_license
Header < boost/svg_plot/svg_fwd.hpp >, 337
boost_license_on
Header < boost/svg_plot/svg.hpp >, 190
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg public member functions, 192
svg_1d_plot public member functions, 205
svg_2d_plot public member functions, 241
svg_boxplot public member functions, 289
box_border
Header < boost/svg_plot/svg_boxplot.hpp >, 280, 319
svg_boxplot public member functions, 289
svg_boxplot_series public member functions, 321
box_fill
Header < boost/svg_plot/svg_boxplot.hpp >, 280, 319
svg_boxplot public member functions, 290
svg_boxplot_series public member functions, 321
box_style
Header < boost/svg_plot/svg_style.hpp >, 344
box_style public member functions
fill, 345
margin, 345
stroke, 345
width, 345
box_width
Header < boost/svg_plot/svg_boxplot.hpp >, 280, 319
svg_boxplot public member functions, 290
svg_boxplot_series public member functions, 322

C

calculate_plot_window
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 206
svg_2d_plot private member functions, 270
svg_boxplot public member functions, 290
calculate_quantiles
Header < boost/svg_plot/svg_boxplot.hpp >, 319
svg_boxplot_series public member functions, 322
calculate_transform
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
svg_1d_plot public member functions, 206
circle_element
Header < boost/svg_plot/detail/svg_tag.hpp >, 113
circle_element public member functions
class_id, 114
clip_id, 114
id, 114
class_id
circle_element public member functions, 114
clip_path_element public member functions, 116
ellipse_element public member functions, 120
g_element public member functions, 123
Header < boost/svg_plot/detail/svg_tag.hpp >, 113, 116, 119, 122, 129, 133, 140, 144, 148, 151, 155, 159, 166
line_element public member functions, 130
path_element public member functions, 134
polygon_element public member functions, 141
polyline_element public member functions, 145
qurve_element public member functions, 149
rect_element public member functions, 152

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

svg_element public member functions, 156
text_element public member functions, 160
tspan_element public member functions, 167
clear
g_element public member functions, 123
Header < boost/svg_plot/detail/svg_tag.hpp >, 122
clear_all
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot protected member functions, 223
svg_2d_plot protected member functions, 271
svg_boxplot public member functions, 290
clear_background
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot protected member functions, 223
svg_2d_plot protected member functions, 272
svg_boxplot protected member functions, 316
clear_grids
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot protected member functions, 223
svg_2d_plot protected member functions, 272
svg_boxplot protected member functions, 316
clear_legend
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot protected member functions, 223
svg_2d_plot protected member functions, 272
svg_boxplot protected member functions, 316
clear_plot_background
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot protected member functions, 223
svg_2d_plot protected member functions, 272
svg_boxplot protected member functions, 316
clear_points
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot protected member functions, 224
svg_2d_plot protected member functions, 272
svg_boxplot protected member functions, 316
clear_title
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot protected member functions, 224
svg_2d_plot protected member functions, 272
svg_boxplot protected member functions, 316
clear_x_axis
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot protected member functions, 224
svg_2d_plot protected member functions, 272
svg_boxplot protected member functions, 316
clear_y_axis
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot protected member functions, 224
svg_2d_plot protected member functions, 272
svg_boxplot protected member functions, 316
clip_id
circle_element public member functions, 114
clip_path_element public member functions, 117
ellipse_element public member functions, 120
g_element public member functions, 123
Header < boost/svg_plot/detail/svg_tag.hpp >, 113, 116, 119, 122, 129, 133, 140, 144, 148, 151, 155, 159, 166
line_element public member functions, 130
path_element public member functions, 134
polygon_element public member functions, 141
polyline_element public member functions, 145
qurve_element public member functions, 149
rect_element public member functions, 152
svg_element public member functions, 156
text_element public member functions, 161
tspan_element public member functions, 167
clip_path_element
Header < boost/svg_plot/detail/svg_tag.hpp >, 116
clip_path_element public member functions
class_id, 116
clip_id, 117
id, 117
close_to
Header < boost/svg_plot/detail/FP_compare.hpp >, 106
close_to public member functions
size, 106
strength, 106
color
axis_line_style public member functions, 341
Header < boost/svg_plot/svg_style.hpp >, 340
constant_to_rgb
Header < boost/svg_plot/svg_color.hpp >, 327, 332
coord_precision
Header < boost/svg_plot/svg.hpp >, 190
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg public member functions, 193
svg_1d_plot public member functions, 206
svg_2d_plot public member functions, 241
svg_boxplot public member functions, 290
coord_precison
Header < boost/svg_plot/svg_fwd.hpp >, 337
copyright_date
Header < boost/svg_plot/svg.hpp >, 190
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Header < boost/svg_plot/svg_fwd.hpp >, 337
svg public member functions, 193
svg_1d_plot public member functions, 206
svg_2d_plot public member functions, 241
svg_boxplot public member functions, 290
copyright_holder
Header < boost/svg_plot/svg.hpp >, 190
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
Header < boost/svg_plot/svg_fwd.hpp >, 337
svg public member functions, 193
svg_1d_plot public member functions, 206
svg_2d_plot public member functions, 241
svg_boxplot public member functions, 290

D

data_lines_width
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 206
svg_2d_plot public member functions, 242
svg_boxplot public member functions, 291
default_plot_point_style
Header < boost/svg_plot/svg_style.hpp >, 338
deg_free
Header < boost/svg_plot/uncertain.hpp >, 371
unc public member functions, 372
Demonstration of using 1D data that includes information about its Uncertainty
width, 52
Demonstration of using 1D data that includes information about its uncertainty
autoscale, 74
autoscale_check_limits, 74
autoscale_plusminus, 74
Description
my_color, 336
p, 176
p0, 177
r, 179
t, 178
ts, 367
description
Header < boost/svg_plot/svg.hpp >, 190
svg public member functions, 193
document_size
Header < boost/svg_plot/svg.hpp >, 190
svg public member functions, 194
document_title
Header < boost/svg_plot/svg.hpp >, 190
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
Header < boost/svg_plot/svg_fwd.hpp >, 337
svg public member functions, 194
svg_1d_plot public member functions, 206
svg_2d_plot public member functions, 242

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

svg_boxplot public member functions, 291
draw_axes
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
svg_1d_plot public member functions, 206
draw_bars
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot private member functions, 270
draw_histogram
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot private member functions, 270
draw_legend
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot protected member functions, 224
svg_2d_plot protected member functions, 272
svg_boxplot protected member functions, 316
draw_plot_lines
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot private member functions, 271
draw_plot_points
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot private member functions, 271
draw_title
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot protected member functions, 224
svg_2d_plot protected member functions, 273
svg_boxplot public member functions, 292
draw_x_axis
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot protected member functions, 225
svg_2d_plot protected member functions, 273
svg_boxplot public member functions, 292
draw_x_axis_label
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot protected member functions, 225
svg_2d_plot protected member functions, 273
svg_boxplot public member functions, 292
draw_y_axis
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_2d_plot private member functions, 271
svg_boxplot public member functions, 292
draw_y_axis_label
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_2d_plot private member functions, 271
svg_boxplot public member functions, 292
dx
Header < boost/svg_plot/detail/svg_tag.hpp >, 166
tspan_element public member functions, 168
dy

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Header < boost/svg_plot/detail/svg_tag.hpp >, 166
tspan_element public member functions, 168

E

ellipse_element
Header < boost/svg_plot/detail/svg_tag.hpp >, 119
ellipse_element public member functions
class_id, 120
clip_id, 120
id, 120
epsilon
Header < boost/svg_plot/detail/FP_compare.hpp >, 105
extreme_outlier_color
Header < boost/svg_plot/svg_boxplot.hpp >, 280, 319
svg_boxplot public member functions, 293
svg_boxplot_series public member functions, 322
extreme_outlier_fill
Header < boost/svg_plot/svg_boxplot.hpp >, 280, 319
svg_boxplot public member functions, 293
svg_boxplot_series public member functions, 323
extreme_outlier_shape
Header < boost/svg_plot/svg_boxplot.hpp >, 280, 319
svg_boxplot public member functions, 293
svg_boxplot_series public member functions, 323
extreme_outlier_size
Header < boost/svg_plot/svg_boxplot.hpp >, 280, 319
svg_boxplot public member functions, 293
svg_boxplot_series public member functions, 323
extreme_outlier_values_on
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_boxplot public member functions, 293

F

fill
box_style public member functions, 345
Header < boost/svg_plot/svg_style.hpp >, 344
fill_color
Header < boost/svg_plot/detail/svg_tag.hpp >, 166
Header < boost/svg_plot/svg_1d_plot.hpp >, 226
Header < boost/svg_plot/svg_fwd.hpp >, 337
svg_1d_plot_series public member functions, 227
tspan_element public member functions, 168
fill_on
Header < boost/svg_plot/detail/svg_tag.hpp >, 133, 166
Header < boost/svg_plot/svg_fwd.hpp >, 337
path_element public member functions, 134
tspan_element public member functions, 168
font_size
Header < boost/svg_plot/detail/svg_tag.hpp >, 166
tspan_element public member functions, 168
fpt_abs
Header < boost/svg_plot/detail/FP_compare.hpp >, 105

G

g_element
Header < boost/svg_plot/detail/svg_tag.hpp >, 122
g_element public member functions

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

class_id, 123
clear, 123
clip_id, 123
id, 124
push_back, 125
size, 126

H

Header < boost/svg_plot/detail/auto_axes.hpp >
isfinite, 87, 89
mnmx, 87, 90
rounddown10, 87, 93
rounddown2, 87, 94
rounddown5, 87
roundup10, 87, 95
roundup2, 87, 96
roundup5, 87, 97
scale_axis, 87, 100
show, 87, 101
Header < boost/svg_plot/detail/FP_compare.hpp >
close_to, 106
epsilon, 105
fpt_abs, 105
max_value, 105
min_value, 105
safe_fpt_division, 105
size, 106, 107
smallest, 107
strength, 106
Header < boost/svg_plot/detail/svg_tag.hpp >
alignment, 159
circle_element, 113
class_id, 113, 116, 119, 122, 129, 133, 140, 144, 148, 151, 155, 159, 166
clear, 122
clip_id, 113, 116, 119, 122, 129, 133, 140, 144, 148, 151, 155, 159, 166
clip_path_element, 116
dx, 166
dy, 166
ellipse_element, 119
fill_color, 166
fill_on, 133, 166
font_size, 166
g_element, 122
id, 113, 116, 119, 122, 129, 133, 140, 144, 148, 151, 155, 159, 166
line_element, 129
path_element, 133
polyline_element, 144
push_back, 122
qurve_element, 148
rect_element, 151
rotation, 166
size, 122
svg, 113, 116, 119, 122, 129, 133, 144, 148, 151, 159, 164, 166
svg_element, 113, 116, 119, 122, 129, 133, 144, 148, 151, 155, 159, 166
text, 159, 166
text_element, 159
text_element_text, 164

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

text_length, 166
text_parent, 164, 165, 166
tspan_element, 166
use_style, 166
x, 166
y, 166
Header < boost/svg_plot/show_1d_settings.hpp >
l_or_r, 182
t_or_b, 182
Header < boost/svg_plot/svg.hpp >
author, 190
boost_license_on, 190
coord_precision, 190
copyright_date, 190
copyright_holder, 190
description, 190
document_size, 190
document_title, 190
image_filename, 190
license, 190
license_on, 190
size, 190
svg, 190
x_size, 190
y_size, 190
Header < boost/svg_plot/svg_1d_plot.hpp >
autoscale, 198
autoscale_check_limits, 198
autoscale_plusminus, 198
axes_on, 198
background_border_color, 198
background_border_width, 198
background_color, 198
bezier_on, 226
boost_license_on, 198
calculate_plot_window, 198
calculate_transform, 198
clear_all, 198
clear_background, 198
clear_grids, 198
clear_legend, 198
clear_plot_background, 198
clear_points, 198
clear_title, 198
clear_x_axis, 198
clear_y_axis, 198
coord_precision, 198
copyright_date, 198
copyright_holder, 198
data_lines_width, 198
document_title, 198
draw_axes, 198
draw_legend, 198
draw_title, 198
draw_x_axis, 198
draw_x_axis_label, 198
fill_color, 226
image_border_margin, 198

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

image_border_width, 198
image_x_size, 198
image_y_size, 198
legend_background_color, 198
legend_border_color, 198
legend_box_fill_on, 198
legend_color, 198
legend_header_font_size, 198
legend_lines, 198
legend_on, 198
legend_outside, 198
legend_place, 198
legend_title, 198
legend_title_font_size, 198
legend_width, 198
license_attribution, 198
license_commercialuse, 198
license_distribution, 198
license_on, 198
license_reproduction, 198
limit_color, 198
limit_fill_color, 198
line_on, 226
line_width, 226
one_sd_color, 198
place_legend_box, 198
plot_background_color, 198
plot_border_color, 198
plot_border_width, 198
plot_window_on, 198
plot_window_x_left, 198
plot_window_x_right, 198
plot_window_y_bottom, 198
plot_window_y_top, 198
series_count, 226
series_limits_count, 226
set_ids, 198
shape, 226
size, 226
size_legend_box, 198
strip_e0s, 197, 230
stroke_color, 226
svg_1d_plot, 198
svg_1d_plot_series, 226
symbols, 226
three_sd_color, 198
title, 198
title_color, 198
title_font_alignment, 198
title_font_rotation, 198
title_font_size, 198
title_on, 198
two_sd_color, 198
update_image, 198
x_autoscale, 198
x_auto_max_value, 198
x_auto_min_value, 198
x_auto_ticks, 198

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

x_auto_tick_interval, 198
x_axis_color, 198
x_axis_label_color, 198
x_axis_on, 198
x_axis_position, 198
x_axis_width, 198
x_df_color, 198
x_df_on, 198
x_label, 198
x_label_color, 198
x_label_font_size, 198
x_label_on, 198
x_label_units, 198
x_label_units_on, 198
x_label_width, 198
x_major_grid_color, 198
x_major_grid_on, 198
x_major_grid_width, 198
x_major_interval, 198
x_major_labels_side, 198
x_major_label_rotation, 198
x_major_tick, 198
x_major_tick_color, 198
x_major_tick_length, 198
x_major_tick_width, 198
x_max, 198
x_min, 198
x_minor_grid_color, 198
x_minor_grid_on, 198
x_minor_grid_width, 198
x_minor_interval, 198
x_minor_tick_color, 198
x_minor_tick_length, 198
x_minor_tick_width, 198
x_min_ticks, 198
x_num_minor_ticks, 198
x_plusminus_color, 198
x_plusminus_on, 198
x_prefix, 198
x_separator, 198
x_steps, 198
x_suffix, 198
x_ticks_down_on, 198
x_ticks_on_window_or_axis, 198
x_ticks_up_on, 198
x_ticks_values_color, 198
x_ticks_values_font_size, 198
x_ticks_values_ioflags, 198
x_ticks_values_precision, 198
x_tight, 198
x_values_color, 198
x_values_font_size, 198
x_values_ioflags, 198
x_values_on, 198
x_values_precision, 198
x_values_rotation, 198
x_value_font_size, 198
x_value_ioflags, 198

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

x_value_precision, 198
x_with_zero, 198
y_axis_color, 198
y_axis_on, 198
y_label, 198
y_labels_strip_e0s, 198
y_label_color, 198
y_label_units, 198
y_minor_interval, 198
Header < boost/svg_plot/svg_2d_plot.hpp >
autoscale, 231
autoscale_check_limits, 231
autoscale_plusminus, 231
axes_on, 231
background_border_color, 231
background_border_width, 231
background_color, 231
bar_opt, 275
bar_width, 275
bezier_on, 275
boost_license_on, 231
calculate_plot_window, 231
clear_all, 231
clear_background, 231
clear_grids, 231
clear_legend, 231
clear_plot_background, 231
clear_points, 231
clear_title, 231
clear_x_axis, 231
clear_y_axis, 231
coord_precision, 231
copyright_date, 231
copyright_holder, 231
data_lines_width, 231
document_title, 231
draw_bars, 231
draw_histogram, 231
draw_legend, 231
draw_plot_lines, 231
draw_plot_points, 231
draw_title, 231
draw_x_axis, 231
draw_x_axis_label, 231
draw_y_axis, 231
draw_y_axis_label, 231
image_border_margin, 231
image_border_width, 231
image_x_size, 231
image_y_size, 231
legend_background_color, 231
legend_border_color, 231
legend_box_fill_on, 231
legend_color, 231
legend_header_font_size, 231
legend_lines, 231
legend_on, 231
legend_outside, 231

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

legend_place, 231
legend_title, 231
legend_title_font_size, 231
legend_width, 231
license_attribution, 231
license_commercialuse, 231
license_distribution, 231
license_on, 231
license_reproduction, 231
limits_count, 275
limit_color, 231
limit_fill_color, 231
line_on, 275
line_style, 275
line_width, 275
one_sd_color, 231
place_legend_box, 231
plot_background_color, 231
plot_border_color, 231
plot_border_width, 231
plot_window_on, 231
plot_window_x_left, 231
plot_window_x_right, 231
plot_window_y_bottom, 231
plot_window_y_top, 231
set_ids, 231
shape, 275
size, 275
size_legend_box, 231
svg_2d_plot, 231
svg_2d_plot_series, 275
three_sd_color, 231
title, 231
title_color, 231
title_font_alignment, 231
title_font_rotation, 231
title_font_size, 231
title_on, 231
two_sd_color, 231
update_image, 231
values_count, 275
xy_autoscale, 231
xy_values_on, 231
x_autoscale, 231
x_auto_max_value, 231
x_auto_min_value, 231
x_auto_ticks, 231
x_auto_tick_interval, 231
x_axis_color, 231
x_axis_label_color, 231
x_axis_on, 231
x_axis_position, 231
x_axis_width, 231
x_df_color, 231
x_df_on, 231
x_label, 231
x_label_color, 231
x_label_font_size, 231

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

x_label_on, 231
x_label_units, 231
x_label_units_on, 231
x_label_width, 231
x_major_grid_color, 231
x_major_grid_on, 231
x_major_grid_width, 231
x_major_interval, 231
x_major_labels_side, 231
x_major_label_rotation, 231
x_major_tick, 231
x_major_tick_color, 231
x_major_tick_length, 231
x_major_tick_width, 231
x_max, 231
x_min, 231
x_minor_grid_color, 231
x_minor_grid_on, 231
x_minor_grid_width, 231
x_minor_interval, 231
x_minor_tick_color, 231
x_minor_tick_length, 231
x_minor_tick_width, 231
x_min_ticks, 231
x_num_minor_ticks, 231
x_plusminus_color, 231
x_plusminus_on, 231
x_prefix, 231
x_separator, 231
x_steps, 231
x_suffix, 231
x_ticks_down_on, 231
x_ticks_on_window_or_axis, 231
x_ticks_up_on, 231
x_ticks_values_color, 231
x_ticks_values_font_size, 231
x_ticks_values_ioflags, 231
x_ticks_values_precision, 231
x_tight, 231
x_values_color, 231
x_values_font_size, 231
x_values_ioflags, 231
x_values_on, 231
x_values_precision, 231
x_values_rotation, 231
x_value_font_size, 231
x_value_ioflags, 231
x_value_precision, 231
x_with_zero, 231
y_autoscale, 231
y_axis_color, 231
y_axis_label_color, 231
y_axis_on, 231
y_axis_position, 231
y_axis_value_color, 231
y_axis_width, 231
y_df_color, 231
y_df_on, 231

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

y_label, 231
y_labels_strip_e0s, 231
y_label_axis, 231
y_label_color, 231
y_label_font_size, 231
y_label_on, 231
y_label_units, 231
y_label_units_on, 231
y_label_width, 231
y_major_grid_color, 231
y_major_grid_on, 231
y_major_grid_width, 231
y_major_interval, 231
y_major_labels_side, 231
y_major_label_rotation, 231
y_major_tick_color, 231
y_major_tick_length, 231
y_major_tick_width, 231
y_max, 231
y_min, 231
y_minor_grid_color, 231
y_minor_grid_on, 231
y_minor_grid_width, 231
y_minor_interval, 231
y_minor_tick_color, 231
y_minor_tick_length, 231
y_minor_tick_width, 231
y_num_minor_ticks, 231
y_plusminus_color, 231
y_plusminus_on, 231
y_prefix, 231
y_separator, 231
y_suffix, 231
y_ticks_left_on, 231
y_ticks_on_window_or_axis, 231
y_ticks_right_on, 231
y_ticks_values_color, 231
y_ticks_values_font_size, 231
y_ticks_values_ioflags, 231
y_ticks_values_precision, 231
y_values_color, 231
y_values_font_size, 231
y_values_ioflags, 231
y_values_on, 231
y_values_precision, 231
y_values_rotation, 231
y_value_ioflags, 231
y_value_precision, 231
Header < boost/svg_plot/svg_boxplot.hpp >
autoscale, 280
autoscale_check_limits, 280
autoscale_plusminus, 280
axes_on, 280
axis_color, 280, 319
axis_width, 280, 319
background_border_color, 280
background_border_width, 280
background_color, 280

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

boost_license_on, 280
box_border, 280, 319
box_fill, 280, 319
box_width, 280, 319
calculate_plot_window, 280
calculate_quantiles, 319
clear_all, 280
clear_background, 280
clear_grids, 280
clear_legend, 280
clear_plot_background, 280
clear_points, 280
clear_title, 280
clear_x_axis, 280
clear_y_axis, 280
coord_precision, 280
copyright_date, 280
copyright_holder, 280
data_lines_width, 280
document_title, 280
draw_legend, 280
draw_title, 280
draw_x_axis, 280
draw_x_axis_label, 280
draw_y_axis, 280
draw_y_axis_label, 280
extreme_outlier_color, 280, 319
extreme_outlier_fill, 280, 319
extreme_outlier_shape, 280, 319
extreme_outlier_size, 280, 319
extreme_outlier_values_on, 280
image_border_margin, 280
image_border_width, 280
image_x_size, 280
image_y_size, 280
legend_background_color, 280
legend_border_color, 280
legend_box_fill_on, 280
legend_color, 280
legend_header_font_size, 280
legend_lines, 280
legend_on, 280
legend_outside, 280
legend_place, 280
legend_title, 280
legend_title_font_size, 280
legend_width, 280
license_attribution, 280
license_commercialuse, 280
license_distribution, 280
license_on, 280
license_reproduction, 280
limit_color, 280
limit_fill_color, 280
max_whisker_color, 319
max_whisker_width, 319
median_color, 280, 319
median_values_on, 280

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

median_width, 280, 319
min_whisker_color, 319
min_whisker_width, 319
one_sd_color, 280
outlier_color, 280, 319
outlier_fill, 280, 319
outlier_shape, 280, 319
outlier_size, 280, 319
outlier_values_on, 280
place_legend_box, 280
plot_background_color, 280
plot_border_color, 280
plot_border_width, 280
plot_window_on, 280
plot_window_x_left, 280
plot_window_x_right, 280
plot_window_y_bottom, 280
plot_window_y_top, 280
quartile_definition, 280, 319
size_legend_box, 280
svg_boxplot, 280
svg_boxplot_series, 319
three_sd_color, 280
title, 280, 319
title_color, 280
title_font_alignment, 280
title_font_rotation, 280
title_font_size, 280
title_on, 280
two_sd_color, 280
update_image, 280
whisker_length, 280, 319
x_autoscale, 280
x_auto_max_value, 280
x_auto_min_value, 280
x_auto_ticks, 280
x_auto_tick_interval, 280
x_axis_color, 280
x_axis_label_color, 280
x_axis_on, 280
x_axis_position, 280
x_axis_width, 280
x_df_color, 280
x_df_on, 280
x_label, 280
x_label_color, 280
x_label_font_size, 280
x_label_on, 280
x_label_text, 280
x_label_units, 280
x_label_units_on, 280
x_label_width, 280
x_major_grid_color, 280
x_major_grid_on, 280
x_major_grid_width, 280
x_major_interval, 280
x_major_labels, 280
x_major_labels_side, 280

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

x_major_label_rotation, 280
x_major_tick, 280
x_major_tick_color, 280
x_major_tick_length, 280
x_major_tick_width, 280
x_max, 280
x_min, 280
x_minor_grid_color, 280
x_minor_grid_on, 280
x_minor_grid_width, 280
x_minor_interval, 280
x_minor_tick_color, 280
x_minor_tick_length, 280
x_minor_tick_width, 280
x_min_ticks, 280
x_num_minor_ticks, 280
x_plusminus_color, 280
x_plusminus_on, 280
x_prefix, 280
x_separator, 280
x_steps, 280
x_suffix, 280
x_ticks_down_on, 280
x_ticks_on_window_or_axis, 280
x_ticks_up_on, 280
x_ticks_values_color, 280
x_ticks_values_font_size, 280
x_ticks_values_ioflags, 280
x_ticks_values_precision, 280
x_tick_color, 280
x_tick_length, 280
x_tight, 280
x_values_color, 280
x_values_font_size, 280
x_values_ioflags, 280
x_values_on, 280
x_values_precision, 280
x_values_rotation, 280
x_value_font_size, 280
x_value_ioflags, 280
x_value_precision, 280
x_with_zero, 280
y_autoscale, 280
y_axis_color, 280
y_axis_on, 280
y_axis_position, 280
y_label, 280
y_labels_strip_e0s, 280
y_label_color, 280
y_label_text, 280
y_label_units, 280
y_minor_interval, 280
Header < boost/svg_plot/svg_color.hpp >
constant_to_rgb, 327, 332
svg_color, 328
Header < boost/svg_plot/svg_fwd.hpp >
author, 337
boost_license, 337

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

coord_precison, 337
copyright_date, 337
copyright_holder, 337
document_title, 337
fill_color, 337
fill_on, 337
id, 337
is_license, 337
stroke_color, 337
stroke_on, 337
stroke_width, 337
width_on, 337
x_size, 337
y_size, 337
Header < boost/svg_plot/svg_style.hpp >
axis_line_style, 340
bar_opt, 342
bar_style, 342
box_style, 344
color, 340
default_plot_point_style, 338
fill, 344
histogram, 346
histogram_style, 346
label_length, 358
longest_label, 358
margin, 344
plot_line_style, 347
plot_point_style, 349
position, 340
shape, 349
size, 349
stroke, 344
svg_style, 352
text_style, 355
ticks_labels_style, 358
value_style, 361
width, 340, 342, 344, 347
Header < boost/svg_plot/uncertain.hpp >
deg_free, 371
types, 371
unc, 371
uncertainty, 371
unc_of, 370, 376, 377
value, 371
value_of, 370, 380, 381
histogram
Header < boost/svg_plot/svg_style.hpp >, 346
histogram_style public member functions, 346
histogram_style
Header < boost/svg_plot/svg_style.hpp >, 346
histogram_style public member functions
histogram, 346

I

id
circle_element public member functions, 114

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

clip_path_element public member functions, 117
ellipse_element public member functions, 120
g_element public member functions, 124
Header < boost/svg_plot/detail/svg_tag.hpp >, 113, 116, 119, 122, 129, 133, 140, 144, 148, 151, 155, 159, 166
Header < boost/svg_plot/svg_fwd.hpp >, 337
line_element public member functions, 130
path_element public member functions, 135
polygon_element public member functions, 142
polyline_element public member functions, 145
qurve_element public member functions, 149
rect_element public member functions, 152
svg_element public member functions, 156
text_element public member functions, 161
tspan_element public member functions, 169
image_border_margin
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 207
svg_2d_plot public member functions, 242
svg_boxplot public member functions, 294
image_border_width
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 207
svg_2d_plot public member functions, 242
svg_boxplot public member functions, 294
image_filename
Header < boost/svg_plot/svg.hpp >, 190
svg public member functions, 194
image_x_size
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 207
svg_2d_plot public member functions, 243
svg_boxplot public member functions, 294
image_y_size
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 207
svg_2d_plot public member functions, 243
svg_boxplot public member functions, 294
Implementation & Rationale
author, 384
license_attribution, 384
license_distribution, 384
license_reproduction, 384
isfinite
Header < boost/svg_plot/detail/auto_axes.hpp >, 87, 89
is_license
Header < boost/svg_plot/svg_fwd.hpp >, 337

L

label_length

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Header < boost/svg_plot/svg_style.hpp >, 358
ticks_labels_style public member functions, 359
legend_background_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 207
svg_2d_plot public member functions, 243
svg_boxplot public member functions, 294
legend_border_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 208
svg_2d_plot public member functions, 243
svg_boxplot public member functions, 294
legend_box_fill_on
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 208
svg_2d_plot public member functions, 243
svg_boxplot public member functions, 294
legend_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 208
svg_2d_plot public member functions, 243
svg_boxplot public member functions, 295
legend_header_font_size
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 208
svg_2d_plot public member functions, 244
svg_boxplot public member functions, 295
legend_lines
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 208
svg_2d_plot public member functions, 244
svg_boxplot public member functions, 295
legend_on
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 208
svg_2d_plot public member functions, 244
svg_boxplot public member functions, 295
legend_outside
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 209
svg_2d_plot public member functions, 244
svg_boxplot public member functions, 295

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

legend_place
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 209
svg_2d_plot public member functions, 244
svg_boxplot public member functions, 295
legend_title
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 209
svg_2d_plot public member functions, 244
svg_boxplot public member functions, 295
legend_title_font_size
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 209
svg_2d_plot public member functions, 244
svg_boxplot public member functions, 296
legend_width
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 209
svg_2d_plot public member functions, 245
svg_boxplot public member functions, 296
license
Header < boost/svg_plot/svg.hpp >, 190
svg public member functions, 194
license_attribution
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
Implementation & Rationale, 384
svg_1d_plot public member functions, 209
svg_2d_plot public member functions, 245
svg_boxplot public member functions, 296
license_commercialuse
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 209
svg_2d_plot public member functions, 245
svg_boxplot public member functions, 296
license_distribution
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
Implementation & Rationale, 384
svg_1d_plot public member functions, 209
svg_2d_plot public member functions, 245
svg_boxplot public member functions, 296
license_on
Header < boost/svg_plot/svg.hpp >, 190
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg public member functions, 194, 195
svg_1d_plot public member functions, 210
svg_2d_plot public member functions, 245
svg_boxplot public member functions, 296
license_reproduction
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
Implementation & Rationale, 384
svg_1d_plot public member functions, 210
svg_2d_plot public member functions, 245
svg_boxplot public member functions, 296
limits_count
Header < boost/svg_plot/svg_2d_plot.hpp >, 275
svg_2d_plot_series public member functions, 277
limit_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 210
svg_2d_plot public member functions, 245
svg_boxplot public member functions, 296
limit_fill_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 210
svg_2d_plot public member functions, 245
svg_boxplot public member functions, 297
line_element
Header < boost/svg_plot/detail/svg_tag.hpp >, 129
line_element public member functions
class_id, 130
clip_id, 130
id, 130
line_on
Header < boost/svg_plot/svg_1d_plot.hpp >, 226
Header < boost/svg_plot/svg_2d_plot.hpp >, 275
svg_1d_plot_series public member functions, 227
svg_2d_plot_series public member functions, 277
line_style
Header < boost/svg_plot/svg_2d_plot.hpp >, 275
svg_2d_plot_series public member functions, 277
line_width
Header < boost/svg_plot/svg_1d_plot.hpp >, 226
Header < boost/svg_plot/svg_2d_plot.hpp >, 275
svg_1d_plot_series public member functions, 227
svg_2d_plot_series public member functions, 278
longest_label
Header < boost/svg_plot/svg_style.hpp >, 358
ticks_labels_style public member functions, 359
l_or_r
Header < boost/svg_plot/show_1d_settings.hpp >, 182

M

margin

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

box_style public member functions, 345
Header < boost/svg_plot/svg_style.hpp >, 344
max_value
Header < boost/svg_plot/detail/FP_compare.hpp >, 105
max_whisker_color
Header < boost/svg_plot/svg_boxplot.hpp >, 319
svg_boxplot_series public member functions, 323
max_whisker_width
Header < boost/svg_plot/svg_boxplot.hpp >, 319
svg_boxplot_series public member functions, 323
median_color
Header < boost/svg_plot/svg_boxplot.hpp >, 280, 319
svg_boxplot public member functions, 297
svg_boxplot_series public member functions, 324
median_values_on
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_boxplot public member functions, 297
median_width
Header < boost/svg_plot/svg_boxplot.hpp >, 280, 319
svg_boxplot public member functions, 297
svg_boxplot_series public member functions, 324
min_value
Header < boost/svg_plot/detail/FP_compare.hpp >, 105
min_whisker_color
Header < boost/svg_plot/svg_boxplot.hpp >, 319
svg_boxplot_series public member functions, 324
min_whisker_width
Header < boost/svg_plot/svg_boxplot.hpp >, 319
svg_boxplot_series public member functions, 325
mnmx
Header < boost/svg_plot/detail/auto_axes.hpp >, 87, 90
my_blank
svg_color public construct/copy/destruct, 328
my_color
Description, 336

O

one_sd_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 210
svg_2d_plot public member functions, 245
svg_boxplot public member functions, 297
outlier_color
Header < boost/svg_plot/svg_boxplot.hpp >, 280, 319
svg_boxplot public member functions, 297
svg_boxplot_series public member functions, 325
outlier_fill
Header < boost/svg_plot/svg_boxplot.hpp >, 280, 319
svg_boxplot public member functions, 298
svg_boxplot_series public member functions, 325
outlier_shape
Header < boost/svg_plot/svg_boxplot.hpp >, 280, 319
svg_boxplot public member functions, 298
svg_boxplot_series public member functions, 325
outlier_size

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Header < boost/svg_plot/svg_boxplot.hpp >, 280, 319
svg_boxplot public member functions, 298
svg_boxplot_series public member functions, 326
outlier_values_on
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_boxplot public member functions, 298

P

p
Description, 176
polygon_element public member functions, 142
p0
Description, 177
path_element
Header < boost/svg_plot/detail/svg_tag.hpp >, 133
path_element public member functions
class_id, 134
clip_id, 134
fill_on, 134
id, 135
place_legend_box
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot protected member functions, 225
svg_2d_plot protected member functions, 273
svg_boxplot protected member functions, 317
plot_background_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 211
svg_2d_plot public member functions, 246
svg_boxplot public member functions, 299
plot_border_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 211
svg_2d_plot public member functions, 246
svg_boxplot public member functions, 299
plot_border_width
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 211
svg_2d_plot public member functions, 247
svg_boxplot public member functions, 299
plot_line_style
Header < boost/svg_plot/svg_style.hpp >, 347
plot_line_style public member functions
width, 348
plot_point_style
Header < boost/svg_plot/svg_style.hpp >, 349
plot_point_style public member functions
shape, 350
size, 350

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

plot_window_on
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 211
svg_2d_plot public member functions, 247
svg_boxplot public member functions, 299
plot_window_x_left
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 211
svg_2d_plot public member functions, 247
svg_boxplot public member functions, 300
plot_window_x_right
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 211
svg_2d_plot public member functions, 247
svg_boxplot public member functions, 300
plot_window_y_bottom
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 212
svg_2d_plot public member functions, 247
svg_boxplot public member functions, 300
plot_window_y_top
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 212
svg_2d_plot public member functions, 247
svg_boxplot public member functions, 300
polygon_element public member functions
class_id, 141
clip_id, 141
id, 142
p, 142
polyline_element
Header < boost/svg_plot/detail/svg_tag.hpp >, 144
polyline_element public member functions
class_id, 145
clip_id, 145
id, 145
position
axis_line_style public member functions, 341
Header < boost/svg_plot/svg_style.hpp >, 340
push_back
g_element public member functions, 125
Header < boost/svg_plot/detail/svg_tag.hpp >, 122

Q

quartile_definition
Header < boost/svg_plot/svg_boxplot.hpp >, 280, 319
svg_boxplot public member functions, 300

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

svg_boxplot_series public member functions, 326
qurve_element
Header < boost/svg_plot/detail/svg_tag.hpp >, 148
qurve_element public member functions
class_id, 149
clip_id, 149
id, 149

R

r
Description, 179
rect_element
Header < boost/svg_plot/detail/svg_tag.hpp >, 151
rect_element public member functions
class_id, 152
clip_id, 152
id, 152
rotation
Header < boost/svg_plot/detail/svg_tag.hpp >, 166
tspan_element public member functions, 169
rounddown10
Header < boost/svg_plot/detail/auto_axes.hpp >, 87, 93
rounddown2
Header < boost/svg_plot/detail/auto_axes.hpp >, 87, 94
rounddown5
Header < boost/svg_plot/detail/auto_axes.hpp >, 87
roundup10
Header < boost/svg_plot/detail/auto_axes.hpp >, 87, 95
roundup2
Header < boost/svg_plot/detail/auto_axes.hpp >, 87, 96
roundup5
Header < boost/svg_plot/detail/auto_axes.hpp >, 87, 97

S

safe_fpt_division
Header < boost/svg_plot/detail/FP_compare.hpp >, 105
scale_axis
1-D Auto scaling Examples, 29
Header < boost/svg_plot/detail/auto_axes.hpp >, 87, 100
series_count
Header < boost/svg_plot/svg_1d_plot.hpp >, 226
svg_1d_plot_series public member functions, 228
series_limits_count
Header < boost/svg_plot/svg_1d_plot.hpp >, 226
svg_1d_plot_series public member functions, 228
set_ids
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_1d_plot public member functions, 212
svg_2d_plot private member functions, 271
shape
Header < boost/svg_plot/svg_1d_plot.hpp >, 226
Header < boost/svg_plot/svg_2d_plot.hpp >, 275
Header < boost/svg_plot/svg_style.hpp >, 349
plot_point_style public member functions, 350
svg_1d_plot_series public member functions, 228
svg_2d_plot_series public member functions, 278

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

show
Header < boost/svg_plot/detail/auto_axes.hpp >, 87, 101
show_1d_plot_settings
1-D Auto scaling Examples, 29
1-D Data Values Examples, 41
Tutorial: 2D Special Features, 59
size
close_to public member functions, 106
g_element public member functions, 126
Header < boost/svg_plot/detail/FP_compare.hpp >, 106, 107
Header < boost/svg_plot/detail/svg_tag.hpp >, 122
Header < boost/svg_plot/svg.hpp >, 190
Header < boost/svg_plot/svg_1d_plot.hpp >, 226
Header < boost/svg_plot/svg_2d_plot.hpp >, 275
Header < boost/svg_plot/svg_style.hpp >, 349
plot_point_style public member functions, 350
smallest public member functions, 107
svg public member functions, 196
svg_1d_plot_series public member functions, 228
svg_2d_plot_series public member functions, 278
size_legend_box
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot protected member functions, 225
svg_2d_plot protected member functions, 273
svg_boxplot protected member functions, 317
smallest
Header < boost/svg_plot/detail/FP_compare.hpp >, 107
smallest public member functions
size, 107
strength
close_to public member functions, 106
Header < boost/svg_plot/detail/FP_compare.hpp >, 106
strip_e0s
Header < boost/svg_plot/svg_1d_plot.hpp >, 197, 230
stroke
box_style public member functions, 345
Header < boost/svg_plot/svg_style.hpp >, 344
stroke_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 226
Header < boost/svg_plot/svg_fwd.hpp >, 337
svg_1d_plot_series public member functions, 228
stroke_on
Header < boost/svg_plot/svg_fwd.hpp >, 337
stroke_width
Header < boost/svg_plot/svg_fwd.hpp >, 337
svg
Header < boost/svg_plot/detail/svg_tag.hpp >, 113, 116, 119, 122, 129, 133, 144, 148, 151, 159, 164, 166
Header < boost/svg_plot/svg.hpp >, 190
svg public member functions
author, 192
boost_license_on, 192
coord_precision, 193
copyright_date, 193
copyright_holder, 193
description, 193
document_size, 194

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

document_title, 194
image_filename, 194
license, 194
license_on, 194, 195
size, 196
x_size, 196
y_size, 196
svg_1d_plot
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
svg_1d_plot protected member functions
clear_all, 223
clear_background, 223
clear_grids, 223
clear_legend, 223
clear_plot_background, 223
clear_points, 224
clear_title, 224
clear_x_axis, 224
clear_y_axis, 224
draw_legend, 224
draw_title, 224
draw_x_axis, 225
draw_x_axis_label, 225
place_legend_box, 225
size_legend_box, 225
svg_1d_plot public member functions
autoscale, 205
autoscale_check_limits, 205
autoscale_plusminus, 205
axes_on, 205
background_border_color, 205
background_border_width, 205
background_color, 205
boost_license_on, 205
calculate_plot_window, 206
calculate_transform, 206
coord_precision, 206
copyright_date, 206
copyright_holder, 206
data_lines_width, 206
document_title, 206
draw_axes, 206
image_border_margin, 207
image_border_width, 207
image_x_size, 207
image_y_size, 207
legend_background_color, 207
legend_border_color, 208
legend_box_fill_on, 208
legend_color, 208
legend_header_font_size, 208
legend_lines, 208
legend_on, 208
legend_outside, 209
legend_place, 209
legend_title, 209
legend_title_font_size, 209
legend_width, 209

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

license_attribution, 209
license_commercialuse, 209
license_distribution, 209
license_on, 210
license_reproduction, 210
limit_color, 210
limit_fill_color, 210
one_sd_color, 210
plot_background_color, 211
plot_border_color, 211
plot_border_width, 211
plot_window_on, 211
plot_window_x_left, 211
plot_window_x_right, 211
plot_window_y_bottom, 212
plot_window_y_top, 212
set_ids, 212
three_sd_color, 212
title, 212
title_color, 212
title_font_alignment, 212
title_font_rotation, 213
title_font_size, 213
title_on, 213
two_sd_color, 213
update_image, 214
x_autoscale, 214
x_auto_max_value, 214
x_auto_min_value, 214
x_auto_ticks, 214
x_auto_tick_interval, 214
x_axis_color, 214
x_axis_label_color, 215
x_axis_on, 215
x_axis_position, 215
x_axis_width, 215
x_df_color, 215
x_df_on, 215
x_label, 215
x_label_color, 215
x_label_font_size, 216
x_label_on, 216
x_label_units, 216
x_label_units_on, 216
x_label_width, 216
x_major_grid_color, 216
x_major_grid_on, 217
x_major_grid_width, 217
x_major_interval, 217
x_major_labels_side, 217
x_major_label_rotation, 217
x_major_tick, 217
x_major_tick_color, 217
x_major_tick_length, 217
x_major_tick_width, 218
x_max, 218
x_min, 218
x_minor_grid_color, 218

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

x_minor_grid_on, 218
x_minor_grid_width, 218
x_minor_interval, 218
x_minor_tick_color, 219
x_minor_tick_length, 219
x_minor_tick_width, 219
x_min_ticks, 218
x_num_minor_ticks, 219
x_plusminus_color, 219
x_plusminus_on, 219
x_prefix, 219
x_separator, 219
x_steps, 220
x_suffix, 220
x_ticks_down_on, 220
x_ticks_on_window_or_axis, 220
x_ticks_up_on, 220
x_ticks_values_color, 220
x_ticks_values_font_size, 220
x_ticks_values_ioflags, 220
x_ticks_values_precision, 221
x_tight, 221
x_values_color, 221
x_values_font_size, 221
x_values_ioflags, 222
x_values_on, 222
x_values_precision, 222
x_values_rotation, 222
x_value_font_size, 221
x_value_ioflags, 221
x_value_precision, 221
x_with_zero, 222
y_axis_color, 222
y_axis_on, 222
y_label, 222
y_labels_strip_e0s, 223
y_label_color, 223
y_label_units, 223
y_minor_interval, 223
svg_1d_plot_series
Header < boost/svg_plot/svg_1d_plot.hpp >, 226
svg_1d_plot_series public member functions
bezier_on, 227
fill_color, 227
line_on, 227
line_width, 227
series_count, 228
series_limits_count, 228
shape, 228
size, 228
stroke_color, 228
symbols, 229
svg_2d_plot
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot private member functions
calculate_plot_window, 270
draw_bars, 270
draw_histogram, 270

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

draw_plot_lines, 271
draw_plot_points, 271
draw_y_axis, 271
draw_y_axis_label, 271
set_ids, 271
update_image, 271
svg_2d_plot protected member functions
clear_all, 271
clear_background, 272
clear_grids, 272
clear_legend, 272
clear_plot_background, 272
clear_points, 272
clear_title, 272
clear_x_axis, 272
clear_y_axis, 272
draw_legend, 272
draw_title, 273
draw_x_axis, 273
draw_x_axis_label, 273
place_legend_box, 273
size_legend_box, 273
svg_2d_plot public member functions
autoscale, 240
autoscale_check_limits, 240
autoscale_plusminus, 241
axes_on, 241
background_border_color, 241
background_border_width, 241
background_color, 241
boost_license_on, 241
coord_precision, 241
copyright_date, 241
copyright_holder, 241
data_lines_width, 242
document_title, 242
image_border_margin, 242
image_border_width, 242
image_x_size, 243
image_y_size, 243
legend_background_color, 243
legend_border_color, 243
legend_box_fill_on, 243
legend_color, 243
legend_header_font_size, 244
legend_lines, 244
legend_on, 244
legend_outside, 244
legend_place, 244
legend_title, 244
legend_title_font_size, 244
legend_width, 245
license_attribution, 245
license_commercialuse, 245
license_distribution, 245
license_on, 245
license_reproduction, 245
limit_color, 245

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

limit_fill_color, 245
one_sd_color, 245
plot_background_color, 246
plot_border_color, 246
plot_border_width, 247
plot_window_on, 247
plot_window_x_left, 247
plot_window_x_right, 247
plot_window_y_bottom, 247
plot_window_y_top, 247
three_sd_color, 247
title, 247
title_color, 248
title_font_alignment, 248
title_font_rotation, 248
title_font_size, 248
title_on, 249
two_sd_color, 249
xy_autoscale, 258
xy_values_on, 258
x_autoscale, 250
x_auto_max_value, 249
x_auto_min_value, 249
x_auto_ticks, 249
x_auto_tick_interval, 249
x_axis_color, 250
x_axis_label_color, 250
x_axis_on, 250
x_axis_position, 250
x_axis_width, 250
x_df_color, 251
x_df_on, 251
x_label, 251
x_label_color, 251
x_label_font_size, 251
x_label_on, 251
x_label_units, 251
x_label_units_on, 252
x_label_width, 252
x_major_grid_color, 252
x_major_grid_on, 252
x_major_grid_width, 252
x_major_interval, 252
x_major_labels_side, 253
x_major_label_rotation, 252
x_major_tick, 253
x_major_tick_color, 253
x_major_tick_length, 253
x_major_tick_width, 253
x_max, 253
x_min, 253
x_minor_grid_color, 254
x_minor_grid_on, 254
x_minor_grid_width, 254
x_minor_interval, 254
x_minor_tick_color, 254
x_minor_tick_length, 254
x_minor_tick_width, 254

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

x_min_ticks, 253
x_num_minor_ticks, 254
x_plusminus_color, 255
x_plusminus_on, 255
x_prefix, 255
x_separator, 255
x_steps, 255
x_suffix, 255
x_ticks_down_on, 255
x_ticks_on_window_or_axis, 255
x_ticks_up_on, 256
x_ticks_values_color, 256
x_ticks_values_font_size, 256
x_ticks_values_ioflags, 256
x_ticks_values_precision, 256
x_tight, 256
x_values_color, 257
x_values_font_size, 257
x_values_ioflags, 257
x_values_on, 257
x_values_precision, 257
x_values_rotation, 258
x_value_font_size, 256
x_value_ioflags, 257
x_value_precision, 257
x_with_zero, 258
y_autoscale, 259
y_axis_color, 259
y_axis_label_color, 259
y_axis_on, 259
y_axis_position, 259
y_axis_value_color, 259
y_axis_width, 260
y_df_color, 260
y_df_on, 260
y_label, 260
y_labels_strip_e0s, 262
y_label_axis, 260
y_label_color, 261
y_label_font_size, 261
y_label_on, 261
y_label_units, 261
y_label_units_on, 261
y_label_width, 262
y_major_grid_color, 262
y_major_grid_on, 262
y_major_grid_width, 263
y_major_interval, 263
y_major_labels_side, 263
y_major_label_rotation, 263
y_major_tick_color, 264
y_major_tick_length, 264
y_major_tick_width, 264
y_max, 264
y_min, 264
y_minor_grid_color, 264
y_minor_grid_on, 264
y_minor_grid_width, 265

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

y_minor_interval, 265
y_minor_tick_color, 265
y_minor_tick_length, 265
y_minor_tick_width, 265
y_num_minor_ticks, 265
y_plusminus_color, 266
y_plusminus_on, 266
y_prefix, 266
y_separator, 266
y_suffix, 266
y_ticks_left_on, 267
y_ticks_on_window_or_axis, 267
y_ticks_right_on, 267
y_ticks_values_color, 267
y_ticks_values_font_size, 268
y_ticks_values_ioflags, 268
y_ticks_values_precision, 268
y_values_color, 269
y_values_font_size, 269
y_values_ioflags, 269
y_values_on, 269
y_values_precision, 270
y_values_rotation, 270
y_value_ioflags, 268
y_value_precision, 268
svg_2d_plot_series
Header < boost/svg_plot/svg_2d_plot.hpp >, 275
svg_2d_plot_series public member functions
bar_opt, 276
bar_width, 276
bezier_on, 277
limits_count, 277
line_on, 277
line_style, 277
line_width, 278
shape, 278
size, 278
values_count, 278
svg_boxplot
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_boxplot protected member functions
clear_background, 316
clear_grids, 316
clear_legend, 316
clear_plot_background, 316
clear_points, 316
clear_title, 316
clear_x_axis, 316
clear_y_axis, 316
draw_legend, 316
place_legend_box, 317
size_legend_box, 317
svg_boxplot public member functions
autoscale, 288
autoscale_check_limits, 288
autoscale_plusminus, 288
axes_on, 288
axis_color, 288

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

axis_width, 289
background_border_color, 289
background_border_width, 289
background_color, 289
boost_license_on, 289
box_border, 289
box_fill, 290
box_width, 290
calculate_plot_window, 290
clear_all, 290
coord_precision, 290
copyright_date, 290
copyright_holder, 290
data_lines_width, 291
document_title, 291
draw_title, 292
draw_x_axis, 292
draw_x_axis_label, 292
draw_y_axis, 292
draw_y_axis_label, 292
extreme_outlier_color, 293
extreme_outlier_fill, 293
extreme_outlier_shape, 293
extreme_outlier_size, 293
extreme_outlier_values_on, 293
image_border_margin, 294
image_border_width, 294
image_x_size, 294
image_y_size, 294
legend_background_color, 294
legend_border_color, 294
legend_box_fill_on, 294
legend_color, 295
legend_header_font_size, 295
legend_lines, 295
legend_on, 295
legend_outside, 295
legend_place, 295
legend_title, 295
legend_title_font_size, 296
legend_width, 296
license_attribution, 296
license_commercialuse, 296
license_distribution, 296
license_on, 296
license_reproduction, 296
limit_color, 296
limit_fill_color, 297
median_color, 297
median_values_on, 297
median_width, 297
one_sd_color, 297
outlier_color, 297
outlier_fill, 298
outlier_shape, 298
outlier_size, 298
outlier_values_on, 298
plot_background_color, 299

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

plot_border_color, 299
plot_border_width, 299
plot_window_on, 299
plot_window_x_left, 300
plot_window_x_right, 300
plot_window_y_bottom, 300
plot_window_y_top, 300
quartile_definition, 300
three_sd_color, 300
title, 300
title_color, 301
title_font_alignment, 301
title_font_rotation, 301
title_font_size, 301
title_on, 302
two_sd_color, 302
update_image, 302
whisker_length, 303
x_autoscale, 303
x_auto_max_value, 303
x_auto_min_value, 303
x_auto_ticks, 303
x_auto_tick_interval, 303
x_axis_color, 304
x_axis_label_color, 304
x_axis_on, 304
x_axis_position, 304
x_axis_width, 304
x_df_color, 304
x_df_on, 304
x_label, 305
x_label_color, 305
x_label_font_size, 305
x_label_on, 305
x_label_text, 306
x_label_units, 306
x_label_units_on, 306
x_label_width, 306
x_major_grid_color, 306
x_major_grid_on, 306
x_major_grid_width, 306
x_major_interval, 306
x_major_labels, 307
x_major_labels_side, 307
x_major_label_rotation, 307
x_major_tick, 307
x_major_tick_color, 307
x_major_tick_length, 307
x_major_tick_width, 307
x_max, 307
x_min, 308
x_minor_grid_color, 308
x_minor_grid_on, 308
x_minor_grid_width, 308
x_minor_interval, 308
x_minor_tick_color, 308
x_minor_tick_length, 308
x_minor_tick_width, 309

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

x_min_ticks, 308
x_num_minor_ticks, 309
x_plusminus_color, 309
x_plusminus_on, 309
x_prefix, 309
x_separator, 309
x_steps, 309
x_suffix, 309
x_ticks_down_on, 310
x_ticks_on_window_or_axis, 310
x_ticks_up_on, 310
x_ticks_values_color, 310
x_ticks_values_font_size, 311
x_ticks_values_ioflags, 311
x_ticks_values_precision, 311
x_tick_color, 310
x_tick_length, 310
x_tight, 311
x_values_color, 312
x_values_font_size, 312
x_values_ioflags, 312
x_values_on, 312
x_values_precision, 312
x_values_rotation, 312
x_value_font_size, 311
x_value_ioflags, 311
x_value_precision, 311
x_with_zero, 312
y_autoscale, 313
y_axis_color, 313
y_axis_on, 313
y_axis_position, 313
y_label, 314
y_labels_strip_e0s, 314
y_label_color, 314
y_label_text, 314
y_label_units, 314
y_minor_interval, 315
svg_boxplot_series
Header < boost/svg_plot/svg_boxplot.hpp >, 319
svg_boxplot_series public member functions
axis_color, 320
axis_width, 321
box_border, 321
box_fill, 321
box_width, 322
calculate_quantiles, 322
extreme_outlier_color, 322
extreme_outlier_fill, 323
extreme_outlier_shape, 323
extreme_outlier_size, 323
max_whisker_color, 323
max_whisker_width, 323
median_color, 324
median_width, 324
min_whisker_color, 324
min_whisker_width, 325
outlier_color, 325

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

outlier_fill, 325
outlier_shape, 325
outlier_size, 326
quartile_definition, 326
title, 326
whisker_length, 327
svg_color
Header < boost/svg_plot/svg_color.hpp >, 328
svg_color public construct/copy/destruct
my_blank, 328
svg_element
Header < boost/svg_plot/detail/svg_tag.hpp >, 113, 116, 119, 122, 129, 133, 144, 148, 151, 155, 159, 166
svg_element public member functions
class_id, 156
clip_id, 156
id, 156
svg_style
Header < boost/svg_plot/svg_style.hpp >, 352
symbols
Header < boost/svg_plot/svg_1d_plot.hpp >, 226
svg_1d_plot_series public member functions, 229

T

t
Description, 178
text
Header < boost/svg_plot/detail/svg_tag.hpp >, 159, 166
text_element public member functions, 162
tspan_element public member functions, 170
text_element
Header < boost/svg_plot/detail/svg_tag.hpp >, 159
text_element public member functions
alignment, 160
class_id, 160
clip_id, 161
id, 161
text, 162
text_element_text
Header < boost/svg_plot/detail/svg_tag.hpp >, 164
text_length
Header < boost/svg_plot/detail/svg_tag.hpp >, 166
tspan_element public member functions, 170
text_parent
Header < boost/svg_plot/detail/svg_tag.hpp >, 164, 165, 166
text_style
Header < boost/svg_plot/svg_style.hpp >, 355
three_sd_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 212
svg_2d_plot public member functions, 247
svg_boxplot public member functions, 300
ticks_labels_style
Header < boost/svg_plot/svg_style.hpp >, 358
ticks_labels_style public member functions
label_length, 359

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

longest_label, 359
title
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280, 319
svg_1d_plot public member functions, 212
svg_2d_plot public member functions, 247
svg_boxplot public member functions, 300
svg_boxplot_series public member functions, 326
title_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 212
svg_2d_plot public member functions, 248
svg_boxplot public member functions, 301
title_font_alignment
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 212
svg_2d_plot public member functions, 248
svg_boxplot public member functions, 301
title_font_rotation
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 213
svg_2d_plot public member functions, 248
svg_boxplot public member functions, 301
title_font_size
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 213
svg_2d_plot public member functions, 248
svg_boxplot public member functions, 301
title_on
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 213
svg_2d_plot public member functions, 249
svg_boxplot public member functions, 302
ts
Description, 367
tspan_element
Header < boost/svg_plot/detail/svg_tag.hpp >, 166
tspan_element public member functions
class_id, 167
clip_id, 167
dx, 168
dy, 168
fill_color, 168
fill_on, 168
font_size, 168
id, 169
rotation, 169

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

text, 170
text_length, 170
use_style, 170
x, 171
y, 171
Tutorial: 2D Special Features
show_1d_plot_settings, 59
two_sd_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 213
svg_2d_plot public member functions, 249
svg_boxplot public member functions, 302
types
Header < boost/svg_plot/uncertain.hpp >, 371
unc public member functions, 372
t_or_b
Header < boost/svg_plot/show_1d_settings.hpp >, 182

U

unc
Header < boost/svg_plot/uncertain.hpp >, 371
unc public member functions
deg_free, 372
types, 372
uncertainty, 372
value, 372
uncertainty
Header < boost/svg_plot/uncertain.hpp >, 371
unc public member functions, 372
unc_of
Header < boost/svg_plot/uncertain.hpp >, 370, 376, 377
update_image
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 214
svg_2d_plot private member functions, 271
svg_boxplot public member functions, 302
use_style
Header < boost/svg_plot/detail/svg_tag.hpp >, 166
tspan_element public member functions, 170

V

value
Header < boost/svg_plot/uncertain.hpp >, 371
unc public member functions, 372
values_count
Header < boost/svg_plot/svg_2d_plot.hpp >, 275
svg_2d_plot_series public member functions, 278
value_of
Header < boost/svg_plot/uncertain.hpp >, 370, 380, 381
value_style
Header < boost/svg_plot/svg_style.hpp >, 361

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

W

whisker_length
Header < boost/svg_plot/svg_boxplot.hpp >, 280, 319
svg_boxplot public member functions, 303
svg_boxplot_series public member functions, 327
width
axis_line_style public member functions, 341
bar_style public member functions, 343
box_style public member functions, 345
Demonstration of using 1D data that includes information about its Uncertainty, 52
Header < boost/svg_plot/svg_style.hpp >, 340, 342, 344, 347
plot_line_style public member functions, 348
width_on
Header < boost/svg_plot/svg_fwd.hpp >, 337

X

x
Header < boost/svg_plot/detail/svg_tag.hpp >, 166
tspan_element public member functions, 171
xy_autoscale
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 258
xy_values_on
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 258
x_autoscale
1-D Auto scaling Examples, 29
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 214
svg_2d_plot public member functions, 250
svg_boxplot public member functions, 303
x_auto_max_value
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 214
svg_2d_plot public member functions, 249
svg_boxplot public member functions, 303
x_auto_min_value
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 214
svg_2d_plot public member functions, 249
svg_boxplot public member functions, 303
x_auto_ticks
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 214
svg_2d_plot public member functions, 249
svg_boxplot public member functions, 303
x_auto_tick_interval
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 214
svg_2d_plot public member functions, 249
svg_boxplot public member functions, 303
x_axis_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 214
svg_2d_plot public member functions, 250
svg_boxplot public member functions, 304
x_axis_label_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 215
svg_2d_plot public member functions, 250
svg_boxplot public member functions, 304
x_axis_on
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 215
svg_2d_plot public member functions, 250
svg_boxplot public member functions, 304
x_axis_position
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 215
svg_2d_plot public member functions, 250
svg_boxplot public member functions, 304
x_axis_width
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 215
svg_2d_plot public member functions, 250
svg_boxplot public member functions, 304
x_df_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 215
svg_2d_plot public member functions, 251
svg_boxplot public member functions, 304
x_df_on
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 215
svg_2d_plot public member functions, 251
svg_boxplot public member functions, 304
x_label
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 215

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

svg_2d_plot public member functions, 251
svg_boxplot public member functions, 305
x_label_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 215
svg_2d_plot public member functions, 251
svg_boxplot public member functions, 305
x_label_font_size
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 216
svg_2d_plot public member functions, 251
svg_boxplot public member functions, 305
x_label_on
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 216
svg_2d_plot public member functions, 251
svg_boxplot public member functions, 305
x_label_text
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_boxplot public member functions, 306
x_label_units
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 216
svg_2d_plot public member functions, 251
svg_boxplot public member functions, 306
x_label_units_on
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 216
svg_2d_plot public member functions, 252
svg_boxplot public member functions, 306
x_label_width
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 216
svg_2d_plot public member functions, 252
svg_boxplot public member functions, 306
x_major_grid_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 216
svg_2d_plot public member functions, 252
svg_boxplot public member functions, 306
x_major_grid_on
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

svg_1d_plot public member functions, 217
svg_2d_plot public member functions, 252
svg_boxplot public member functions, 306
x_major_grid_width
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 217
svg_2d_plot public member functions, 252
svg_boxplot public member functions, 306
x_major_interval
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 217
svg_2d_plot public member functions, 252
svg_boxplot public member functions, 306
x_major_labels
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_boxplot public member functions, 307
x_major_labels_side
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 217
svg_2d_plot public member functions, 253
svg_boxplot public member functions, 307
x_major_label_rotation
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 217
svg_2d_plot public member functions, 252
svg_boxplot public member functions, 307
x_major_tick
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 217
svg_2d_plot public member functions, 253
svg_boxplot public member functions, 307
x_major_tick_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 217
svg_2d_plot public member functions, 253
svg_boxplot public member functions, 307
x_major_tick_length
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 217
svg_2d_plot public member functions, 253
svg_boxplot public member functions, 307
x_major_tick_width
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 218
svg_2d_plot public member functions, 253
svg_boxplot public member functions, 307
x_max
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 218
svg_2d_plot public member functions, 253
svg_boxplot public member functions, 307
x_min
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 218
svg_2d_plot public member functions, 253
svg_boxplot public member functions, 308
x_minor_grid_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 218
svg_2d_plot public member functions, 254
svg_boxplot public member functions, 308
x_minor_grid_on
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 218
svg_2d_plot public member functions, 254
svg_boxplot public member functions, 308
x_minor_grid_width
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 218
svg_2d_plot public member functions, 254
svg_boxplot public member functions, 308
x_minor_interval
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 218
svg_2d_plot public member functions, 254
svg_boxplot public member functions, 308
x_minor_tick_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 219
svg_2d_plot public member functions, 254
svg_boxplot public member functions, 308
x_minor_tick_length
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 219

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

svg_2d_plot public member functions, 254
svg_boxplot public member functions, 308
x_minor_tick_width
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 219
svg_2d_plot public member functions, 254
svg_boxplot public member functions, 309
x_min_ticks
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 218
svg_2d_plot public member functions, 253
svg_boxplot public member functions, 308
x_num_minor_ticks
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 219
svg_2d_plot public member functions, 254
svg_boxplot public member functions, 309
x_plusminus_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 219
svg_2d_plot public member functions, 255
svg_boxplot public member functions, 309
x_plusminus_on
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 219
svg_2d_plot public member functions, 255
svg_boxplot public member functions, 309
x_prefix
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 219
svg_2d_plot public member functions, 255
svg_boxplot public member functions, 309
x_separator
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 219
svg_2d_plot public member functions, 255
svg_boxplot public member functions, 309
x_size
Header < boost/svg_plot/svg.hpp >, 190
Header < boost/svg_plot/svg_fwd.hpp >, 337
svg public member functions, 196
x_steps
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 220
svg_2d_plot public member functions, 255
svg_boxplot public member functions, 309
x_suffix
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 220
svg_2d_plot public member functions, 255
svg_boxplot public member functions, 309
x_ticks_down_on
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 220
svg_2d_plot public member functions, 255
svg_boxplot public member functions, 310
x_ticks_on_window_or_axis
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 220
svg_2d_plot public member functions, 255
svg_boxplot public member functions, 310
x_ticks_up_on
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 220
svg_2d_plot public member functions, 256
svg_boxplot public member functions, 310
x_ticks_values_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 220
svg_2d_plot public member functions, 256
svg_boxplot public member functions, 310
x_ticks_values_font_size
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 220
svg_2d_plot public member functions, 256
svg_boxplot public member functions, 311
x_ticks_values_ioflags
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 220
svg_2d_plot public member functions, 256
svg_boxplot public member functions, 311
x_ticks_values_precision
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 221

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

svg_2d_plot public member functions, 256
svg_boxplot public member functions, 311
x_tick_color
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_boxplot public member functions, 310
x_tick_length
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_boxplot public member functions, 310
x_tight
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 221
svg_2d_plot public member functions, 256
svg_boxplot public member functions, 311
x_values_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 221
svg_2d_plot public member functions, 257
svg_boxplot public member functions, 312
x_values_font_size
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 221
svg_2d_plot public member functions, 257
svg_boxplot public member functions, 312
x_values_ioflags
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 222
svg_2d_plot public member functions, 257
svg_boxplot public member functions, 312
x_values_on
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 222
svg_2d_plot public member functions, 257
svg_boxplot public member functions, 312
x_values_precision
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 222
svg_2d_plot public member functions, 257
svg_boxplot public member functions, 312
x_values_rotation
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 222
svg_2d_plot public member functions, 258
svg_boxplot public member functions, 312
x_value_font_size

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 221
svg_2d_plot public member functions, 256
svg_boxplot public member functions, 311
x_value_ioflags
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 221
svg_2d_plot public member functions, 257
svg_boxplot public member functions, 311
x_value_precision
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 221
svg_2d_plot public member functions, 257
svg_boxplot public member functions, 311
x_with_zero
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 222
svg_2d_plot public member functions, 258
svg_boxplot public member functions, 312

Y

y
Header < boost/svg_plot/detail/svg_tag.hpp >, 166
tspan_element public member functions, 171
y_autoscale
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_2d_plot public member functions, 259
svg_boxplot public member functions, 313
y_axis_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 222
svg_2d_plot public member functions, 259
svg_boxplot public member functions, 313
y_axis_label_color
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 259
y_axis_on
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 222
svg_2d_plot public member functions, 259
svg_boxplot public member functions, 313
y_axis_position
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

svg_2d_plot public member functions, 259
svg_boxplot public member functions, 313
y_axis_value_color
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 259
y_axis_width
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 260
y_df_color
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 260
y_df_on
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 260
y_label
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 222
svg_2d_plot public member functions, 260
svg_boxplot public member functions, 314
y_labels_strip_e0s
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 223
svg_2d_plot public member functions, 262
svg_boxplot public member functions, 314
y_label_axis
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 260
y_label_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 223
svg_2d_plot public member functions, 261
svg_boxplot public member functions, 314
y_label_font_size
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 261
y_label_on
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 261
y_label_text
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_boxplot public member functions, 314
y_label_units
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 223
svg_2d_plot public member functions, 261
svg_boxplot public member functions, 314
y_label_units_on
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 261
y_label_width

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 262
y_major_grid_color
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 262
y_major_grid_on
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 262
y_major_grid_width
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 263
y_major_interval
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 263
y_major_labels_side
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 263
y_major_label_rotation
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 263
y_major_tick_color
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 264
y_major_tick_length
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 264
y_major_tick_width
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 264
y_max
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 264
y_min
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 264
y_minor_grid_color
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 264
y_minor_grid_on
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 264
y_minor_grid_width
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 265
y_minor_interval
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 223
svg_2d_plot public member functions, 265
svg_boxplot public member functions, 315
y_minor_tick_color
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 265
y_minor_tick_length
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 265
y_minor_tick_width

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 265
y_num_minor_ticks
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 265
y_plusminus_color
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 266
y_plusminus_on
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 266
y_prefix
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 266
y_separator
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 266
y_size
Header < boost/svg_plot/svg.hpp >, 190
Header < boost/svg_plot/svg_fwd.hpp >, 337
svg public member functions, 196
y_suffix
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 266
y_ticks_left_on
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 267
y_ticks_on_window_or_axis
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 267
y_ticks_right_on
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 267
y_ticks_values_color
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 267
y_ticks_values_font_size
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 268
y_ticks_values_ioflags
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 268
y_ticks_values_precision
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 268
y_values_color
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 269
y_values_font_size
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 269
y_values_ioflags
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 269
y_values_on
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 269
y_values_precision

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 270
y_values_rotation
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 270
y_value_ioflags
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 268
y_value_precision
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 268

Function Index

Symbols

1-D Auto scaling Examples
scale_axis, 29
show_1d_plot_settings, 29
x_autoscale, 29
1-D Data Values Examples
show_1d_plot_settings, 41

A

alignment
Header < boost/svg_plot/detail/svg_tag.hpp >, 159
text_element public member functions, 160
author
Header < boost/svg_plot/svg.hpp >, 190
Header < boost/svg_plot/svg_fwd.hpp >, 337
Implementation & Rationale, 384
svg public member functions, 192
autoscale
Demonstration of using 1D data that includes information about its uncertainty, 74
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 205
svg_2d_plot public member functions, 240
svg_boxplot public member functions, 288
autoscale_check_limits
Demonstration of using 1D data that includes information about its uncertainty, 74
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 205
svg_2d_plot public member functions, 240
svg_boxplot public member functions, 288
autoscale_plusminus
Demonstration of using 1D data that includes information about its uncertainty, 74
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 205
svg_2d_plot public member functions, 241
svg_boxplot public member functions, 288
axes_on

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 205
svg_2d_plot public member functions, 241
svg_boxplot public member functions, 288
axis_color
Header < boost/svg_plot/svg_boxplot.hpp >, 280, 319
svg_boxplot public member functions, 288
svg_boxplot_series public member functions, 320
axis_line_style
Header < boost/svg_plot/svg_style.hpp >, 340
axis_line_style public member functions
color, 341
position, 341
width, 341
axis_width
Header < boost/svg_plot/svg_boxplot.hpp >, 280, 319
svg_boxplot public member functions, 289
svg_boxplot_series public member functions, 321

B

background_border_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 205
svg_2d_plot public member functions, 241
svg_boxplot public member functions, 289
background_border_width
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 205
svg_2d_plot public member functions, 241
svg_boxplot public member functions, 289
background_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 205
svg_2d_plot public member functions, 241
svg_boxplot public member functions, 289
bar_opt
bar_style public member functions, 342
Header < boost/svg_plot/svg_2d_plot.hpp >, 275
Header < boost/svg_plot/svg_style.hpp >, 342
svg_2d_plot_series public member functions, 276
bar_style
Header < boost/svg_plot/svg_style.hpp >, 342
bar_style public member functions
bar_opt, 342
width, 343
bar_width
Header < boost/svg_plot/svg_2d_plot.hpp >, 275
svg_2d_plot_series public member functions, 276
bezier_on

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Header < boost/svg_plot/svg_1d_plot.hpp >, 226
Header < boost/svg_plot/svg_2d_plot.hpp >, 275
svg_1d_plot_series public member functions, 227
svg_2d_plot_series public member functions, 277
boost_license
Header < boost/svg_plot/svg_fwd.hpp >, 337
boost_license_on
Header < boost/svg_plot/svg.hpp >, 190
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg public member functions, 192
svg_1d_plot public member functions, 205
svg_2d_plot public member functions, 241
svg_boxplot public member functions, 289
box_border
Header < boost/svg_plot/svg_boxplot.hpp >, 280, 319
svg_boxplot public member functions, 289
svg_boxplot_series public member functions, 321
box_fill
Header < boost/svg_plot/svg_boxplot.hpp >, 280, 319
svg_boxplot public member functions, 290
svg_boxplot_series public member functions, 321
box_style
Header < boost/svg_plot/svg_style.hpp >, 344
box_style public member functions
fill, 345
margin, 345
stroke, 345
width, 345
box_width
Header < boost/svg_plot/svg_boxplot.hpp >, 280, 319
svg_boxplot public member functions, 290
svg_boxplot_series public member functions, 322

C

calculate_plot_window
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 206
svg_2d_plot private member functions, 270
svg_boxplot public member functions, 290
calculate_quantiles
Header < boost/svg_plot/svg_boxplot.hpp >, 319
svg_boxplot_series public member functions, 322
calculate_transform
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
svg_1d_plot public member functions, 206
circle_element
Header < boost/svg_plot/detail/svg_tag.hpp >, 113
circle_element public member functions
class_id, 114
clip_id, 114
id, 114
class_id
circle_element public member functions, 114

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

clip_path_element public member functions, 116
ellipse_element public member functions, 120
g_element public member functions, 123
Header < boost/svg_plot/detail/svg_tag.hpp >, 113, 116, 119, 122, 129, 133, 140, 144, 148, 151, 155, 159, 166
line_element public member functions, 130
path_element public member functions, 134
polygon_element public member functions, 141
polyline_element public member functions, 145
qurve_element public member functions, 149
rect_element public member functions, 152
svg_element public member functions, 156
text_element public member functions, 160
tspan_element public member functions, 167
clear
g_element public member functions, 123
Header < boost/svg_plot/detail/svg_tag.hpp >, 122
clear_all
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot protected member functions, 223
svg_2d_plot protected member functions, 271
svg_boxplot public member functions, 290
clear_background
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot protected member functions, 223
svg_2d_plot protected member functions, 272
svg_boxplot protected member functions, 316
clear_grids
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot protected member functions, 223
svg_2d_plot protected member functions, 272
svg_boxplot protected member functions, 316
clear_legend
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot protected member functions, 223
svg_2d_plot protected member functions, 272
svg_boxplot protected member functions, 316
clear_plot_background
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot protected member functions, 223
svg_2d_plot protected member functions, 272
svg_boxplot protected member functions, 316
clear_points
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot protected member functions, 224
svg_2d_plot protected member functions, 272
svg_boxplot protected member functions, 316

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

clear_title
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot protected member functions, 224
svg_2d_plot protected member functions, 272
svg_boxplot protected member functions, 316
clear_x_axis
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot protected member functions, 224
svg_2d_plot protected member functions, 272
svg_boxplot protected member functions, 316
clear_y_axis
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot protected member functions, 224
svg_2d_plot protected member functions, 272
svg_boxplot protected member functions, 316
clip_id
circle_element public member functions, 114
clip_path_element public member functions, 117
ellipse_element public member functions, 120
g_element public member functions, 123
Header < boost/svg_plot/detail/svg_tag.hpp >, 113, 116, 119, 122, 129, 133, 140, 144, 148, 151, 155, 159, 166
line_element public member functions, 130
path_element public member functions, 134
polygon_element public member functions, 141
polyline_element public member functions, 145
qurve_element public member functions, 149
rect_element public member functions, 152
svg_element public member functions, 156
text_element public member functions, 161
tspan_element public member functions, 167
clip_path_element
Header < boost/svg_plot/detail/svg_tag.hpp >, 116
clip_path_element public member functions
class_id, 116
clip_id, 117
id, 117
close_to
Header < boost/svg_plot/detail/FP_compare.hpp >, 106
close_to public member functions
size, 106
strength, 106
color
axis_line_style public member functions, 341
Header < boost/svg_plot/svg_style.hpp >, 340
constant_to_rgb
Header < boost/svg_plot/svg_color.hpp >, 327, 332
coord_precision
Header < boost/svg_plot/svg.hpp >, 190
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg public member functions, 193

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

svg_1d_plot public member functions, 206
svg_2d_plot public member functions, 241
svg_boxplot public member functions, 290
coord_precison
Header < boost/svg_plot/svg_fwd.hpp >, 337
copyright_date
Header < boost/svg_plot/svg.hpp >, 190
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
Header < boost/svg_plot/svg_fwd.hpp >, 337
svg public member functions, 193
svg_1d_plot public member functions, 206
svg_2d_plot public member functions, 241
svg_boxplot public member functions, 290
copyright_holder
Header < boost/svg_plot/svg.hpp >, 190
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
Header < boost/svg_plot/svg_fwd.hpp >, 337
svg public member functions, 193
svg_1d_plot public member functions, 206
svg_2d_plot public member functions, 241
svg_boxplot public member functions, 290

D

data_lines_width
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 206
svg_2d_plot public member functions, 242
svg_boxplot public member functions, 291
default_plot_point_style
Header < boost/svg_plot/svg_style.hpp >, 338
deg_free
Header < boost/svg_plot/uncertain.hpp >, 371
unc public member functions, 372
Demonstration of using 1D data that includes information about its Uncertainty
width, 52
Demonstration of using 1D data that includes information about its uncertainty
autoscale, 74
autoscale_check_limits, 74
autoscale_plusminus, 74
Description
my_color, 336
p, 176
p0, 177
r, 179
t, 178
ts, 367
description
Header < boost/svg_plot/svg.hpp >, 190
svg public member functions, 193
document_size
Header < boost/svg_plot/svg.hpp >, 190

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

svg public member functions, 194
document_title
Header < boost/svg_plot/svg.hpp >, 190
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
Header < boost/svg_plot/svg_fwd.hpp >, 337
svg public member functions, 194
svg_1d_plot public member functions, 206
svg_2d_plot public member functions, 242
svg_boxplot public member functions, 291
draw_axes
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
svg_1d_plot public member functions, 206
draw_bars
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot private member functions, 270
draw_histogram
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot private member functions, 270
draw_legend
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot protected member functions, 224
svg_2d_plot protected member functions, 272
svg_boxplot protected member functions, 316
draw_plot_lines
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot private member functions, 271
draw_plot_points
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot private member functions, 271
draw_title
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot protected member functions, 224
svg_2d_plot protected member functions, 273
svg_boxplot public member functions, 292
draw_x_axis
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot protected member functions, 225
svg_2d_plot protected member functions, 273
svg_boxplot public member functions, 292
draw_x_axis_label
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot protected member functions, 225
svg_2d_plot protected member functions, 273
svg_boxplot public member functions, 292
draw_y_axis
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_2d_plot private member functions, 271

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

svg_boxplot public member functions, 292
draw_y_axis_label
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_2d_plot private member functions, 271
svg_boxplot public member functions, 292
dx
Header < boost/svg_plot/detail/svg_tag.hpp >, 166
tspan_element public member functions, 168
dy
Header < boost/svg_plot/detail/svg_tag.hpp >, 166
tspan_element public member functions, 168

E

ellipse_element
Header < boost/svg_plot/detail/svg_tag.hpp >, 119
ellipse_element public member functions
class_id, 120
clip_id, 120
id, 120
epsilon
Header < boost/svg_plot/detail/FP_compare.hpp >, 105
extreme_outlier_color
Header < boost/svg_plot/svg_boxplot.hpp >, 280, 319
svg_boxplot public member functions, 293
svg_boxplot_series public member functions, 322
extreme_outlier_fill
Header < boost/svg_plot/svg_boxplot.hpp >, 280, 319
svg_boxplot public member functions, 293
svg_boxplot_series public member functions, 323
extreme_outlier_shape
Header < boost/svg_plot/svg_boxplot.hpp >, 280, 319
svg_boxplot public member functions, 293
svg_boxplot_series public member functions, 323
extreme_outlier_size
Header < boost/svg_plot/svg_boxplot.hpp >, 280, 319
svg_boxplot public member functions, 293
svg_boxplot_series public member functions, 323
extreme_outlier_values_on
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_boxplot public member functions, 293

F

fill
box_style public member functions, 345
Header < boost/svg_plot/svg_style.hpp >, 344
fill_color
Header < boost/svg_plot/detail/svg_tag.hpp >, 166
Header < boost/svg_plot/svg_1d_plot.hpp >, 226
Header < boost/svg_plot/svg_fwd.hpp >, 337
svg_1d_plot_series public member functions, 227
tspan_element public member functions, 168
fill_on
Header < boost/svg_plot/detail/svg_tag.hpp >, 133, 166
Header < boost/svg_plot/svg_fwd.hpp >, 337
path_element public member functions, 134
tspan_element public member functions, 168

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

font_size
Header < boost/svg_plot/detail/svg_tag.hpp >, 166
tspan_element public member functions, 168
fpt_abs
Header < boost/svg_plot/detail/FP_compare.hpp >, 105

G

g_element
Header < boost/svg_plot/detail/svg_tag.hpp >, 122
g_element public member functions
class_id, 123
clear, 123
clip_id, 123
id, 124
push_back, 125
size, 126

H

Header < boost/svg_plot/detail/auto_axes.hpp >
isfinite, 87, 89
mnmx, 87, 90
rounddown10, 87, 93
rounddown2, 87, 94
rounddown5, 87
roundup10, 87, 95
roundup2, 87, 96
roundup5, 87, 97
scale_axis, 87, 100
show, 87, 101
Header < boost/svg_plot/detail/FP_compare.hpp >
close_to, 106
epsilon, 105
fpt_abs, 105
max_value, 105
min_value, 105
safe_fpt_division, 105
size, 106, 107
smallest, 107
strength, 106
Header < boost/svg_plot/detail/svg_tag.hpp >
alignment, 159
circle_element, 113
class_id, 113, 116, 119, 122, 129, 133, 140, 144, 148, 151, 155, 159, 166
clear, 122
clip_id, 113, 116, 119, 122, 129, 133, 140, 144, 148, 151, 155, 159, 166
clip_path_element, 116
dx, 166
dy, 166
ellipse_element, 119
fill_color, 166
fill_on, 133, 166
font_size, 166
g_element, 122
id, 113, 116, 119, 122, 129, 133, 140, 144, 148, 151, 155, 159, 166
line_element, 129
path_element, 133
polyline_element, 144

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

push_back, 122
qurve_element, 148
rect_element, 151
rotation, 166
size, 122
svg, 113, 116, 119, 122, 129, 133, 144, 148, 151, 159, 164, 166
svg_element, 113, 116, 119, 122, 129, 133, 144, 148, 151, 155, 159, 166
text, 159, 166
text_element, 159
text_element_text, 164
text_length, 166
text_parent, 164, 165, 166
tspan_element, 166
use_style, 166
x, 166
y, 166
Header < boost/svg_plot/show_1d_settings.hpp >
l_or_r, 182
t_or_b, 182
Header < boost/svg_plot/svg.hpp >
author, 190
boost_license_on, 190
coord_precision, 190
copyright_date, 190
copyright_holder, 190
description, 190
document_size, 190
document_title, 190
image_filename, 190
license, 190
license_on, 190
size, 190
svg, 190
x_size, 190
y_size, 190
Header < boost/svg_plot/svg_1d_plot.hpp >
autoscale, 198
autoscale_check_limits, 198
autoscale_plusminus, 198
axes_on, 198
background_border_color, 198
background_border_width, 198
background_color, 198
bezier_on, 226
boost_license_on, 198
calculate_plot_window, 198
calculate_transform, 198
clear_all, 198
clear_background, 198
clear_grids, 198
clear_legend, 198
clear_plot_background, 198
clear_points, 198
clear_title, 198
clear_x_axis, 198
clear_y_axis, 198
coord_precision, 198
copyright_date, 198

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

copyright_holder, 198
data_lines_width, 198
document_title, 198
draw_axes, 198
draw_legend, 198
draw_title, 198
draw_x_axis, 198
draw_x_axis_label, 198
fill_color, 226
image_border_margin, 198
image_border_width, 198
image_x_size, 198
image_y_size, 198
legend_background_color, 198
legend_border_color, 198
legend_box_fill_on, 198
legend_color, 198
legend_header_font_size, 198
legend_lines, 198
legend_on, 198
legend_outside, 198
legend_place, 198
legend_title, 198
legend_title_font_size, 198
legend_width, 198
license_attribution, 198
license_commercialuse, 198
license_distribution, 198
license_on, 198
license_reproduction, 198
limit_color, 198
limit_fill_color, 198
line_on, 226
line_width, 226
one_sd_color, 198
place_legend_box, 198
plot_background_color, 198
plot_border_color, 198
plot_border_width, 198
plot_window_on, 198
plot_window_x_left, 198
plot_window_x_right, 198
plot_window_y_bottom, 198
plot_window_y_top, 198
series_count, 226
series_limits_count, 226
set_ids, 198
shape, 226
size, 226
size_legend_box, 198
strip_e0s, 197, 230
stroke_color, 226
svg_1d_plot, 198
svg_1d_plot_series, 226
symbols, 226
three_sd_color, 198
title, 198
title_color, 198

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

title_font_alignment, 198
title_font_rotation, 198
title_font_size, 198
title_on, 198
two_sd_color, 198
update_image, 198
x_autoscale, 198
x_auto_max_value, 198
x_auto_min_value, 198
x_auto_ticks, 198
x_auto_tick_interval, 198
x_axis_color, 198
x_axis_label_color, 198
x_axis_on, 198
x_axis_position, 198
x_axis_width, 198
x_df_color, 198
x_df_on, 198
x_label, 198
x_label_color, 198
x_label_font_size, 198
x_label_on, 198
x_label_units, 198
x_label_units_on, 198
x_label_width, 198
x_major_grid_color, 198
x_major_grid_on, 198
x_major_grid_width, 198
x_major_interval, 198
x_major_labels_side, 198
x_major_label_rotation, 198
x_major_tick, 198
x_major_tick_color, 198
x_major_tick_length, 198
x_major_tick_width, 198
x_max, 198
x_min, 198
x_minor_grid_color, 198
x_minor_grid_on, 198
x_minor_grid_width, 198
x_minor_interval, 198
x_minor_tick_color, 198
x_minor_tick_length, 198
x_minor_tick_width, 198
x_min_ticks, 198
x_num_minor_ticks, 198
x_plusminus_color, 198
x_plusminus_on, 198
x_prefix, 198
x_separator, 198
x_steps, 198
x_suffix, 198
x_ticks_down_on, 198
x_ticks_on_window_or_axis, 198
x_ticks_up_on, 198
x_ticks_values_color, 198
x_ticks_values_font_size, 198
x_ticks_values_ioflags, 198

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

x_ticks_values_precision, 198
x_tight, 198
x_values_color, 198
x_values_font_size, 198
x_values_ioflags, 198
x_values_on, 198
x_values_precision, 198
x_values_rotation, 198
x_value_font_size, 198
x_value_ioflags, 198
x_value_precision, 198
x_with_zero, 198
y_axis_color, 198
y_axis_on, 198
y_label, 198
y_labels_strip_e0s, 198
y_label_color, 198
y_label_units, 198
y_minor_interval, 198
Header < boost/svg_plot/svg_2d_plot.hpp >
autoscale, 231
autoscale_check_limits, 231
autoscale_plusminus, 231
axes_on, 231
background_border_color, 231
background_border_width, 231
background_color, 231
bar_opt, 275
bar_width, 275
bezier_on, 275
boost_license_on, 231
calculate_plot_window, 231
clear_all, 231
clear_background, 231
clear_grids, 231
clear_legend, 231
clear_plot_background, 231
clear_points, 231
clear_title, 231
clear_x_axis, 231
clear_y_axis, 231
coord_precision, 231
copyright_date, 231
copyright_holder, 231
data_lines_width, 231
document_title, 231
draw_bars, 231
draw_histogram, 231
draw_legend, 231
draw_plot_lines, 231
draw_plot_points, 231
draw_title, 231
draw_x_axis, 231
draw_x_axis_label, 231
draw_y_axis, 231
draw_y_axis_label, 231
image_border_margin, 231
image_border_width, 231

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

image_x_size, 231
image_y_size, 231
legend_background_color, 231
legend_border_color, 231
legend_box_fill_on, 231
legend_color, 231
legend_header_font_size, 231
legend_lines, 231
legend_on, 231
legend_outside, 231
legend_place, 231
legend_title, 231
legend_title_font_size, 231
legend_width, 231
license_attribution, 231
license_commercialuse, 231
license_distribution, 231
license_on, 231
license_reproduction, 231
limits_count, 275
limit_color, 231
limit_fill_color, 231
line_on, 275
line_style, 275
line_width, 275
one_sd_color, 231
place_legend_box, 231
plot_background_color, 231
plot_border_color, 231
plot_border_width, 231
plot_window_on, 231
plot_window_x_left, 231
plot_window_x_right, 231
plot_window_y_bottom, 231
plot_window_y_top, 231
set_ids, 231
shape, 275
size, 275
size_legend_box, 231
svg_2d_plot, 231
svg_2d_plot_series, 275
three_sd_color, 231
title, 231
title_color, 231
title_font_alignment, 231
title_font_rotation, 231
title_font_size, 231
title_on, 231
two_sd_color, 231
update_image, 231
values_count, 275
xy_autoscale, 231
xy_values_on, 231
x_autoscale, 231
x_auto_max_value, 231
x_auto_min_value, 231
x_auto_ticks, 231
x_auto_tick_interval, 231

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

x_axis_color, 231
x_axis_label_color, 231
x_axis_on, 231
x_axis_position, 231
x_axis_width, 231
x_df_color, 231
x_df_on, 231
x_label, 231
x_label_color, 231
x_label_font_size, 231
x_label_on, 231
x_label_units, 231
x_label_units_on, 231
x_label_width, 231
x_major_grid_color, 231
x_major_grid_on, 231
x_major_grid_width, 231
x_major_interval, 231
x_major_labels_side, 231
x_major_label_rotation, 231
x_major_tick, 231
x_major_tick_color, 231
x_major_tick_length, 231
x_major_tick_width, 231
x_max, 231
x_min, 231
x_minor_grid_color, 231
x_minor_grid_on, 231
x_minor_grid_width, 231
x_minor_interval, 231
x_minor_tick_color, 231
x_minor_tick_length, 231
x_minor_tick_width, 231
x_min_ticks, 231
x_num_minor_ticks, 231
x_plusminus_color, 231
x_plusminus_on, 231
x_prefix, 231
x_separator, 231
x_steps, 231
x_suffix, 231
x_ticks_down_on, 231
x_ticks_on_window_or_axis, 231
x_ticks_up_on, 231
x_ticks_values_color, 231
x_ticks_values_font_size, 231
x_ticks_values_ioflags, 231
x_ticks_values_precision, 231
x_tight, 231
x_values_color, 231
x_values_font_size, 231
x_values_ioflags, 231
x_values_on, 231
x_values_precision, 231
x_values_rotation, 231
x_value_font_size, 231
x_value_ioflags, 231
x_value_precision, 231

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

x_with_zero, 231
y_autoscale, 231
y_axis_color, 231
y_axis_label_color, 231
y_axis_on, 231
y_axis_position, 231
y_axis_value_color, 231
y_axis_width, 231
y_df_color, 231
y_df_on, 231
y_label, 231
y_labels_strip_e0s, 231
y_label_axis, 231
y_label_color, 231
y_label_font_size, 231
y_label_on, 231
y_label_units, 231
y_label_units_on, 231
y_label_width, 231
y_major_grid_color, 231
y_major_grid_on, 231
y_major_grid_width, 231
y_major_interval, 231
y_major_labels_side, 231
y_major_label_rotation, 231
y_major_tick_color, 231
y_major_tick_length, 231
y_major_tick_width, 231
y_max, 231
y_min, 231
y_minor_grid_color, 231
y_minor_grid_on, 231
y_minor_grid_width, 231
y_minor_interval, 231
y_minor_tick_color, 231
y_minor_tick_length, 231
y_minor_tick_width, 231
y_num_minor_ticks, 231
y_plusminus_color, 231
y_plusminus_on, 231
y_prefix, 231
y_separator, 231
y_suffix, 231
y_ticks_left_on, 231
y_ticks_on_window_or_axis, 231
y_ticks_right_on, 231
y_ticks_values_color, 231
y_ticks_values_font_size, 231
y_ticks_values_ioflags, 231
y_ticks_values_precision, 231
y_values_color, 231
y_values_font_size, 231
y_values_ioflags, 231
y_values_on, 231
y_values_precision, 231
y_values_rotation, 231
y_value_ioflags, 231
y_value_precision, 231

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Header < boost/svg_plot/svg_boxplot.hpp >
autoscale, 280
autoscale_check_limits, 280
autoscale_plusminus, 280
axes_on, 280
axis_color, 280, 319
axis_width, 280, 319
background_border_color, 280
background_border_width, 280
background_color, 280
boost_license_on, 280
box_border, 280, 319
box_fill, 280, 319
box_width, 280, 319
calculate_plot_window, 280
calculate_quantiles, 319
clear_all, 280
clear_background, 280
clear_grids, 280
clear_legend, 280
clear_plot_background, 280
clear_points, 280
clear_title, 280
clear_x_axis, 280
clear_y_axis, 280
coord_precision, 280
copyright_date, 280
copyright_holder, 280
data_lines_width, 280
document_title, 280
draw_legend, 280
draw_title, 280
draw_x_axis, 280
draw_x_axis_label, 280
draw_y_axis, 280
draw_y_axis_label, 280
extreme_outlier_color, 280, 319
extreme_outlier_fill, 280, 319
extreme_outlier_shape, 280, 319
extreme_outlier_size, 280, 319
extreme_outlier_values_on, 280
image_border_margin, 280
image_border_width, 280
image_x_size, 280
image_y_size, 280
legend_background_color, 280
legend_border_color, 280
legend_box_fill_on, 280
legend_color, 280
legend_header_font_size, 280
legend_lines, 280
legend_on, 280
legend_outside, 280
legend_place, 280
legend_title, 280
legend_title_font_size, 280
legend_width, 280
license_attribution, 280

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

license_commercialuse, 280
license_distribution, 280
license_on, 280
license_reproduction, 280
limit_color, 280
limit_fill_color, 280
max_whisker_color, 319
max_whisker_width, 319
median_color, 280, 319
median_values_on, 280
median_width, 280, 319
min_whisker_color, 319
min_whisker_width, 319
one_sd_color, 280
outlier_color, 280, 319
outlier_fill, 280, 319
outlier_shape, 280, 319
outlier_size, 280, 319
outlier_values_on, 280
place_legend_box, 280
plot_background_color, 280
plot_border_color, 280
plot_border_width, 280
plot_window_on, 280
plot_window_x_left, 280
plot_window_x_right, 280
plot_window_y_bottom, 280
plot_window_y_top, 280
quartile_definition, 280, 319
size_legend_box, 280
svg_boxplot, 280
svg_boxplot_series, 319
three_sd_color, 280
title, 280, 319
title_color, 280
title_font_alignment, 280
title_font_rotation, 280
title_font_size, 280
title_on, 280
two_sd_color, 280
update_image, 280
whisker_length, 280, 319
x_autoscale, 280
x_auto_max_value, 280
x_auto_min_value, 280
x_auto_ticks, 280
x_auto_tick_interval, 280
x_axis_color, 280
x_axis_label_color, 280
x_axis_on, 280
x_axis_position, 280
x_axis_width, 280
x_df_color, 280
x_df_on, 280
x_label, 280
x_label_color, 280
x_label_font_size, 280
x_label_on, 280

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

x_label_text, 280
x_label_units, 280
x_label_units_on, 280
x_label_width, 280
x_major_grid_color, 280
x_major_grid_on, 280
x_major_grid_width, 280
x_major_interval, 280
x_major_labels, 280
x_major_labels_side, 280
x_major_label_rotation, 280
x_major_tick, 280
x_major_tick_color, 280
x_major_tick_length, 280
x_major_tick_width, 280
x_max, 280
x_min, 280
x_minor_grid_color, 280
x_minor_grid_on, 280
x_minor_grid_width, 280
x_minor_interval, 280
x_minor_tick_color, 280
x_minor_tick_length, 280
x_minor_tick_width, 280
x_min_ticks, 280
x_num_minor_ticks, 280
x_plusminus_color, 280
x_plusminus_on, 280
x_prefix, 280
x_separator, 280
x_steps, 280
x_suffix, 280
x_ticks_down_on, 280
x_ticks_on_window_or_axis, 280
x_ticks_up_on, 280
x_ticks_values_color, 280
x_ticks_values_font_size, 280
x_ticks_values_ioflags, 280
x_ticks_values_precision, 280
x_tick_color, 280
x_tick_length, 280
x_tight, 280
x_values_color, 280
x_values_font_size, 280
x_values_ioflags, 280
x_values_on, 280
x_values_precision, 280
x_values_rotation, 280
x_value_font_size, 280
x_value_ioflags, 280
x_value_precision, 280
x_with_zero, 280
y_autoscale, 280
y_axis_color, 280
y_axis_on, 280
y_axis_position, 280
y_label, 280
y_labels_strip_e0s, 280

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

y_label_color, 280
y_label_text, 280
y_label_units, 280
y_minor_interval, 280
Header < boost/svg_plot/svg_color.hpp >
constant_to_rgb, 327, 332
svg_color, 328
Header < boost/svg_plot/svg_fwd.hpp >
author, 337
boost_license, 337
coord_precison, 337
copyright_date, 337
copyright_holder, 337
document_title, 337
fill_color, 337
fill_on, 337
id, 337
is_license, 337
stroke_color, 337
stroke_on, 337
stroke_width, 337
width_on, 337
x_size, 337
y_size, 337
Header < boost/svg_plot/svg_style.hpp >
axis_line_style, 340
bar_opt, 342
bar_style, 342
box_style, 344
color, 340
default_plot_point_style, 338
fill, 344
histogram, 346
histogram_style, 346
label_length, 358
longest_label, 358
margin, 344
plot_line_style, 347
plot_point_style, 349
position, 340
shape, 349
size, 349
stroke, 344
svg_style, 352
text_style, 355
ticks_labels_style, 358
value_style, 361
width, 340, 342, 344, 347
Header < boost/svg_plot/uncertain.hpp >
deg_free, 371
types, 371
unc, 371
uncertainty, 371
unc_of, 370, 376, 377
value, 371
value_of, 370, 380, 381
histogram
Header < boost/svg_plot/svg_style.hpp >, 346

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

histogram_style public member functions, 346
histogram_style
Header < boost/svg_plot/svg_style.hpp >, 346
histogram_style public member functions
histogram, 346

I

id
circle_element public member functions, 114
clip_path_element public member functions, 117
ellipse_element public member functions, 120
g_element public member functions, 124
Header < boost/svg_plot/detail/svg_tag.hpp >, 113, 116, 119, 122, 129, 133, 140, 144, 148, 151, 155, 159, 166
Header < boost/svg_plot/svg_fwd.hpp >, 337
line_element public member functions, 130
path_element public member functions, 135
polygon_element public member functions, 142
polyline_element public member functions, 145
qurve_element public member functions, 149
rect_element public member functions, 152
svg_element public member functions, 156
text_element public member functions, 161
tspan_element public member functions, 169
image_border_margin
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 207
svg_2d_plot public member functions, 242
svg_boxplot public member functions, 294
image_border_width
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 207
svg_2d_plot public member functions, 242
svg_boxplot public member functions, 294
image_filename
Header < boost/svg_plot/svg.hpp >, 190
svg public member functions, 194
image_x_size
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 207
svg_2d_plot public member functions, 243
svg_boxplot public member functions, 294
image_y_size
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 207
svg_2d_plot public member functions, 243
svg_boxplot public member functions, 294
Implementation & Rationale
author, 384
license_attribution, 384

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

license_distribution, 384
license_reproduction, 384
isfinite
Header < boost/svg_plot/detail/auto_axes.hpp >, 87, 89
is_license
Header < boost/svg_plot/svg_fwd.hpp >, 337

L

label_length
Header < boost/svg_plot/svg_style.hpp >, 358
ticks_labels_style public member functions, 359
legend_background_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 207
svg_2d_plot public member functions, 243
svg_boxplot public member functions, 294
legend_border_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 208
svg_2d_plot public member functions, 243
svg_boxplot public member functions, 294
legend_box_fill_on
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 208
svg_2d_plot public member functions, 243
svg_boxplot public member functions, 294
legend_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 208
svg_2d_plot public member functions, 243
svg_boxplot public member functions, 295
legend_header_font_size
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 208
svg_2d_plot public member functions, 244
svg_boxplot public member functions, 295
legend_lines
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 208
svg_2d_plot public member functions, 244
svg_boxplot public member functions, 295
legend_on
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

svg_1d_plot public member functions, 208
svg_2d_plot public member functions, 244
svg_boxplot public member functions, 295
legend_outside
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 209
svg_2d_plot public member functions, 244
svg_boxplot public member functions, 295
legend_place
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 209
svg_2d_plot public member functions, 244
svg_boxplot public member functions, 295
legend_title
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 209
svg_2d_plot public member functions, 244
svg_boxplot public member functions, 295
legend_title_font_size
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 209
svg_2d_plot public member functions, 244
svg_boxplot public member functions, 296
legend_width
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 209
svg_2d_plot public member functions, 245
svg_boxplot public member functions, 296
license
Header < boost/svg_plot/svg.hpp >, 190
svg public member functions, 194
license_attribution
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
Implementation & Rationale, 384
svg_1d_plot public member functions, 209
svg_2d_plot public member functions, 245
svg_boxplot public member functions, 296
license_commercialuse
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 209
svg_2d_plot public member functions, 245
svg_boxplot public member functions, 296
license_distribution
Header < boost/svg_plot/svg_1d_plot.hpp >, 198

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
Implementation & Rationale, 384
svg_1d_plot public member functions, 209
svg_2d_plot public member functions, 245
svg_boxplot public member functions, 296
license_on
Header < boost/svg_plot/svg.hpp >, 190
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg public member functions, 194, 195
svg_1d_plot public member functions, 210
svg_2d_plot public member functions, 245
svg_boxplot public member functions, 296
license_reproduction
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
Implementation & Rationale, 384
svg_1d_plot public member functions, 210
svg_2d_plot public member functions, 245
svg_boxplot public member functions, 296
limits_count
Header < boost/svg_plot/svg_2d_plot.hpp >, 275
svg_2d_plot_series public member functions, 277
limit_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 210
svg_2d_plot public member functions, 245
svg_boxplot public member functions, 296
limit_fill_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 210
svg_2d_plot public member functions, 245
svg_boxplot public member functions, 297
line_element
Header < boost/svg_plot/detail/svg_tag.hpp >, 129
line_element public member functions
class_id, 130
clip_id, 130
id, 130
line_on
Header < boost/svg_plot/svg_1d_plot.hpp >, 226
Header < boost/svg_plot/svg_2d_plot.hpp >, 275
svg_1d_plot_series public member functions, 227
svg_2d_plot_series public member functions, 277
line_style
Header < boost/svg_plot/svg_2d_plot.hpp >, 275
svg_2d_plot_series public member functions, 277
line_width
Header < boost/svg_plot/svg_1d_plot.hpp >, 226
Header < boost/svg_plot/svg_2d_plot.hpp >, 275
svg_1d_plot_series public member functions, 227

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

svg_2d_plot_series public member functions, 278
longest_label
Header < boost/svg_plot/svg_style.hpp >, 358
ticks_labels_style public member functions, 359
l_or_r
Header < boost/svg_plot/show_1d_settings.hpp >, 182

M

margin
box_style public member functions, 345
Header < boost/svg_plot/svg_style.hpp >, 344
max_value
Header < boost/svg_plot/detail/FP_compare.hpp >, 105
max_whisker_color
Header < boost/svg_plot/svg_boxplot.hpp >, 319
svg_boxplot_series public member functions, 323
max_whisker_width
Header < boost/svg_plot/svg_boxplot.hpp >, 319
svg_boxplot_series public member functions, 323
median_color
Header < boost/svg_plot/svg_boxplot.hpp >, 280, 319
svg_boxplot public member functions, 297
svg_boxplot_series public member functions, 324
median_values_on
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_boxplot public member functions, 297
median_width
Header < boost/svg_plot/svg_boxplot.hpp >, 280, 319
svg_boxplot public member functions, 297
svg_boxplot_series public member functions, 324
min_value
Header < boost/svg_plot/detail/FP_compare.hpp >, 105
min_whisker_color
Header < boost/svg_plot/svg_boxplot.hpp >, 319
svg_boxplot_series public member functions, 324
min_whisker_width
Header < boost/svg_plot/svg_boxplot.hpp >, 319
svg_boxplot_series public member functions, 325
mnmx
Header < boost/svg_plot/detail/auto_axes.hpp >, 87, 90
my_blank
svg_color public construct/copy/destruct, 328
my_color
Description, 336

O

one_sd_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 210
svg_2d_plot public member functions, 245
svg_boxplot public member functions, 297
outlier_color
Header < boost/svg_plot/svg_boxplot.hpp >, 280, 319
svg_boxplot public member functions, 297
svg_boxplot_series public member functions, 325

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

outlier_fill
Header < boost/svg_plot/svg_boxplot.hpp >, 280, 319
svg_boxplot public member functions, 298
svg_boxplot_series public member functions, 325
outlier_shape
Header < boost/svg_plot/svg_boxplot.hpp >, 280, 319
svg_boxplot public member functions, 298
svg_boxplot_series public member functions, 325
outlier_size
Header < boost/svg_plot/svg_boxplot.hpp >, 280, 319
svg_boxplot public member functions, 298
svg_boxplot_series public member functions, 326
outlier_values_on
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_boxplot public member functions, 298

P

p
Description, 176
polygon_element public member functions, 142
p0
Description, 177
path_element
Header < boost/svg_plot/detail/svg_tag.hpp >, 133
path_element public member functions
class_id, 134
clip_id, 134
fill_on, 134
id, 135
place_legend_box
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot protected member functions, 225
svg_2d_plot protected member functions, 273
svg_boxplot protected member functions, 317
plot_background_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 211
svg_2d_plot public member functions, 246
svg_boxplot public member functions, 299
plot_border_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 211
svg_2d_plot public member functions, 246
svg_boxplot public member functions, 299
plot_border_width
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 211
svg_2d_plot public member functions, 247
svg_boxplot public member functions, 299

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

plot_line_style
Header < boost/svg_plot/svg_style.hpp >, 347
plot_line_style public member functions
width, 348
plot_point_style
Header < boost/svg_plot/svg_style.hpp >, 349
plot_point_style public member functions
shape, 350
size, 350
plot_window_on
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 211
svg_2d_plot public member functions, 247
svg_boxplot public member functions, 299
plot_window_x_left
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 211
svg_2d_plot public member functions, 247
svg_boxplot public member functions, 300
plot_window_x_right
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 211
svg_2d_plot public member functions, 247
svg_boxplot public member functions, 300
plot_window_y_bottom
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 212
svg_2d_plot public member functions, 247
svg_boxplot public member functions, 300
plot_window_y_top
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 212
svg_2d_plot public member functions, 247
svg_boxplot public member functions, 300
polygon_element public member functions
class_id, 141
clip_id, 141
id, 142
p, 142
polyline_element
Header < boost/svg_plot/detail/svg_tag.hpp >, 144
polyline_element public member functions
class_id, 145
clip_id, 145
id, 145
position
axis_line_style public member functions, 341
Header < boost/svg_plot/svg_style.hpp >, 340

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

push_back
g_element public member functions, 125
Header < boost/svg_plot/detail/svg_tag.hpp >, 122

Q

quartile_definition
Header < boost/svg_plot/svg_boxplot.hpp >, 280, 319
svg_boxplot public member functions, 300
svg_boxplot_series public member functions, 326
qurve_element
Header < boost/svg_plot/detail/svg_tag.hpp >, 148
qurve_element public member functions
class_id, 149
clip_id, 149
id, 149

R

r
Description, 179
rect_element
Header < boost/svg_plot/detail/svg_tag.hpp >, 151
rect_element public member functions
class_id, 152
clip_id, 152
id, 152
rotation
Header < boost/svg_plot/detail/svg_tag.hpp >, 166
tspan_element public member functions, 169
rounddown10
Header < boost/svg_plot/detail/auto_axes.hpp >, 87, 93
rounddown2
Header < boost/svg_plot/detail/auto_axes.hpp >, 87, 94
rounddown5
Header < boost/svg_plot/detail/auto_axes.hpp >, 87
roundup10
Header < boost/svg_plot/detail/auto_axes.hpp >, 87, 95
roundup2
Header < boost/svg_plot/detail/auto_axes.hpp >, 87, 96
roundup5
Header < boost/svg_plot/detail/auto_axes.hpp >, 87, 97

S

safe_fpt_division
Header < boost/svg_plot/detail/FP_compare.hpp >, 105
scale_axis
1-D Auto scaling Examples, 29
Header < boost/svg_plot/detail/auto_axes.hpp >, 87, 100
series_count
Header < boost/svg_plot/svg_1d_plot.hpp >, 226
svg_1d_plot_series public member functions, 228
series_limits_count
Header < boost/svg_plot/svg_1d_plot.hpp >, 226
svg_1d_plot_series public member functions, 228
set_ids
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_1d_plot public member functions, 212

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

svg_2d_plot private member functions, 271
shape
Header < boost/svg_plot/svg_1d_plot.hpp >, 226
Header < boost/svg_plot/svg_2d_plot.hpp >, 275
Header < boost/svg_plot/svg_style.hpp >, 349
plot_point_style public member functions, 350
svg_1d_plot_series public member functions, 228
svg_2d_plot_series public member functions, 278
show
Header < boost/svg_plot/detail/auto_axes.hpp >, 87, 101
show_1d_plot_settings
1-D Auto scaling Examples, 29
1-D Data Values Examples, 41
Tutorial: 2D Special Features, 59
size
close_to public member functions, 106
g_element public member functions, 126
Header < boost/svg_plot/detail/FP_compare.hpp >, 106, 107
Header < boost/svg_plot/detail/svg_tag.hpp >, 122
Header < boost/svg_plot/svg.hpp >, 190
Header < boost/svg_plot/svg_1d_plot.hpp >, 226
Header < boost/svg_plot/svg_2d_plot.hpp >, 275
Header < boost/svg_plot/svg_style.hpp >, 349
plot_point_style public member functions, 350
smallest public member functions, 107
svg public member functions, 196
svg_1d_plot_series public member functions, 228
svg_2d_plot_series public member functions, 278
size_legend_box
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot protected member functions, 225
svg_2d_plot protected member functions, 273
svg_boxplot protected member functions, 317
smallest
Header < boost/svg_plot/detail/FP_compare.hpp >, 107
smallest public member functions
size, 107
strength
close_to public member functions, 106
Header < boost/svg_plot/detail/FP_compare.hpp >, 106
strip_e0s
Header < boost/svg_plot/svg_1d_plot.hpp >, 197, 230
stroke
box_style public member functions, 345
Header < boost/svg_plot/svg_style.hpp >, 344
stroke_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 226
Header < boost/svg_plot/svg_fwd.hpp >, 337
svg_1d_plot_series public member functions, 228
stroke_on
Header < boost/svg_plot/svg_fwd.hpp >, 337
stroke_width
Header < boost/svg_plot/svg_fwd.hpp >, 337
svg
Header < boost/svg_plot/detail/svg_tag.hpp >, 113, 116, 119, 122, 129, 133, 144, 148, 151, 159, 164, 166
Header < boost/svg_plot/svg.hpp >, 190

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

svg public member functions
author, 192
boost_license_on, 192
coord_precision, 193
copyright_date, 193
copyright_holder, 193
description, 193
document_size, 194
document_title, 194
image_filename, 194
license, 194
license_on, 194, 195
size, 196
x_size, 196
y_size, 196
svg_1d_plot
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
svg_1d_plot protected member functions
clear_all, 223
clear_background, 223
clear_grids, 223
clear_legend, 223
clear_plot_background, 223
clear_points, 224
clear_title, 224
clear_x_axis, 224
clear_y_axis, 224
draw_legend, 224
draw_title, 224
draw_x_axis, 225
draw_x_axis_label, 225
place_legend_box, 225
size_legend_box, 225
svg_1d_plot public member functions
autoscale, 205
autoscale_check_limits, 205
autoscale_plusminus, 205
axes_on, 205
background_border_color, 205
background_border_width, 205
background_color, 205
boost_license_on, 205
calculate_plot_window, 206
calculate_transform, 206
coord_precision, 206
copyright_date, 206
copyright_holder, 206
data_lines_width, 206
document_title, 206
draw_axes, 206
image_border_margin, 207
image_border_width, 207
image_x_size, 207
image_y_size, 207
legend_background_color, 207
legend_border_color, 208
legend_box_fill_on, 208
legend_color, 208

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

legend_header_font_size, 208
legend_lines, 208
legend_on, 208
legend_outside, 209
legend_place, 209
legend_title, 209
legend_title_font_size, 209
legend_width, 209
license_attribution, 209
license_commercialuse, 209
license_distribution, 209
license_on, 210
license_reproduction, 210
limit_color, 210
limit_fill_color, 210
one_sd_color, 210
plot_background_color, 211
plot_border_color, 211
plot_border_width, 211
plot_window_on, 211
plot_window_x_left, 211
plot_window_x_right, 211
plot_window_y_bottom, 212
plot_window_y_top, 212
set_ids, 212
three_sd_color, 212
title, 212
title_color, 212
title_font_alignment, 212
title_font_rotation, 213
title_font_size, 213
title_on, 213
two_sd_color, 213
update_image, 214
x_autoscale, 214
x_auto_max_value, 214
x_auto_min_value, 214
x_auto_ticks, 214
x_auto_tick_interval, 214
x_axis_color, 214
x_axis_label_color, 215
x_axis_on, 215
x_axis_position, 215
x_axis_width, 215
x_df_color, 215
x_df_on, 215
x_label, 215
x_label_color, 215
x_label_font_size, 216
x_label_on, 216
x_label_units, 216
x_label_units_on, 216
x_label_width, 216
x_major_grid_color, 216
x_major_grid_on, 217
x_major_grid_width, 217
x_major_interval, 217
x_major_labels_side, 217

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

x_major_label_rotation, 217
x_major_tick, 217
x_major_tick_color, 217
x_major_tick_length, 217
x_major_tick_width, 218
x_max, 218
x_min, 218
x_minor_grid_color, 218
x_minor_grid_on, 218
x_minor_grid_width, 218
x_minor_interval, 218
x_minor_tick_color, 219
x_minor_tick_length, 219
x_minor_tick_width, 219
x_min_ticks, 218
x_num_minor_ticks, 219
x_plusminus_color, 219
x_plusminus_on, 219
x_prefix, 219
x_separator, 219
x_steps, 220
x_suffix, 220
x_ticks_down_on, 220
x_ticks_on_window_or_axis, 220
x_ticks_up_on, 220
x_ticks_values_color, 220
x_ticks_values_font_size, 220
x_ticks_values_ioflags, 220
x_ticks_values_precision, 221
x_tight, 221
x_values_color, 221
x_values_font_size, 221
x_values_ioflags, 222
x_values_on, 222
x_values_precision, 222
x_values_rotation, 222
x_value_font_size, 221
x_value_ioflags, 221
x_value_precision, 221
x_with_zero, 222
y_axis_color, 222
y_axis_on, 222
y_label, 222
y_labels_strip_e0s, 223
y_label_color, 223
y_label_units, 223
y_minor_interval, 223
svg_1d_plot_series
Header < boost/svg_plot/svg_1d_plot.hpp >, 226
svg_1d_plot_series public member functions
bezier_on, 227
fill_color, 227
line_on, 227
line_width, 227
series_count, 228
series_limits_count, 228
shape, 228
size, 228

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

stroke_color, 228
symbols, 229
svg_2d_plot
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot private member functions
calculate_plot_window, 270
draw_bars, 270
draw_histogram, 270
draw_plot_lines, 271
draw_plot_points, 271
draw_y_axis, 271
draw_y_axis_label, 271
set_ids, 271
update_image, 271
svg_2d_plot protected member functions
clear_all, 271
clear_background, 272
clear_grids, 272
clear_legend, 272
clear_plot_background, 272
clear_points, 272
clear_title, 272
clear_x_axis, 272
clear_y_axis, 272
draw_legend, 272
draw_title, 273
draw_x_axis, 273
draw_x_axis_label, 273
place_legend_box, 273
size_legend_box, 273
svg_2d_plot public member functions
autoscale, 240
autoscale_check_limits, 240
autoscale_plusminus, 241
axes_on, 241
background_border_color, 241
background_border_width, 241
background_color, 241
boost_license_on, 241
coord_precision, 241
copyright_date, 241
copyright_holder, 241
data_lines_width, 242
document_title, 242
image_border_margin, 242
image_border_width, 242
image_x_size, 243
image_y_size, 243
legend_background_color, 243
legend_border_color, 243
legend_box_fill_on, 243
legend_color, 243
legend_header_font_size, 244
legend_lines, 244
legend_on, 244
legend_outside, 244
legend_place, 244
legend_title, 244

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

legend_title_font_size, 244
legend_width, 245
license_attribution, 245
license_commercialuse, 245
license_distribution, 245
license_on, 245
license_reproduction, 245
limit_color, 245
limit_fill_color, 245
one_sd_color, 245
plot_background_color, 246
plot_border_color, 246
plot_border_width, 247
plot_window_on, 247
plot_window_x_left, 247
plot_window_x_right, 247
plot_window_y_bottom, 247
plot_window_y_top, 247
three_sd_color, 247
title, 247
title_color, 248
title_font_alignment, 248
title_font_rotation, 248
title_font_size, 248
title_on, 249
two_sd_color, 249
xy_autoscale, 258
xy_values_on, 258
x_autoscale, 250
x_auto_max_value, 249
x_auto_min_value, 249
x_auto_ticks, 249
x_auto_tick_interval, 249
x_axis_color, 250
x_axis_label_color, 250
x_axis_on, 250
x_axis_position, 250
x_axis_width, 250
x_df_color, 251
x_df_on, 251
x_label, 251
x_label_color, 251
x_label_font_size, 251
x_label_on, 251
x_label_units, 251
x_label_units_on, 252
x_label_width, 252
x_major_grid_color, 252
x_major_grid_on, 252
x_major_grid_width, 252
x_major_interval, 252
x_major_labels_side, 253
x_major_label_rotation, 252
x_major_tick, 253
x_major_tick_color, 253
x_major_tick_length, 253
x_major_tick_width, 253
x_max, 253

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

x_min, 253
x_minor_grid_color, 254
x_minor_grid_on, 254
x_minor_grid_width, 254
x_minor_interval, 254
x_minor_tick_color, 254
x_minor_tick_length, 254
x_minor_tick_width, 254
x_min_ticks, 253
x_num_minor_ticks, 254
x_plusminus_color, 255
x_plusminus_on, 255
x_prefix, 255
x_separator, 255
x_steps, 255
x_suffix, 255
x_ticks_down_on, 255
x_ticks_on_window_or_axis, 255
x_ticks_up_on, 256
x_ticks_values_color, 256
x_ticks_values_font_size, 256
x_ticks_values_ioflags, 256
x_ticks_values_precision, 256
x_tight, 256
x_values_color, 257
x_values_font_size, 257
x_values_ioflags, 257
x_values_on, 257
x_values_precision, 257
x_values_rotation, 258
x_value_font_size, 256
x_value_ioflags, 257
x_value_precision, 257
x_with_zero, 258
y_autoscale, 259
y_axis_color, 259
y_axis_label_color, 259
y_axis_on, 259
y_axis_position, 259
y_axis_value_color, 259
y_axis_width, 260
y_df_color, 260
y_df_on, 260
y_label, 260
y_labels_strip_e0s, 262
y_label_axis, 260
y_label_color, 261
y_label_font_size, 261
y_label_on, 261
y_label_units, 261
y_label_units_on, 261
y_label_width, 262
y_major_grid_color, 262
y_major_grid_on, 262
y_major_grid_width, 263
y_major_interval, 263
y_major_labels_side, 263
y_major_label_rotation, 263

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

y_major_tick_color, 264
y_major_tick_length, 264
y_major_tick_width, 264
y_max, 264
y_min, 264
y_minor_grid_color, 264
y_minor_grid_on, 264
y_minor_grid_width, 265
y_minor_interval, 265
y_minor_tick_color, 265
y_minor_tick_length, 265
y_minor_tick_width, 265
y_num_minor_ticks, 265
y_plusminus_color, 266
y_plusminus_on, 266
y_prefix, 266
y_separator, 266
y_suffix, 266
y_ticks_left_on, 267
y_ticks_on_window_or_axis, 267
y_ticks_right_on, 267
y_ticks_values_color, 267
y_ticks_values_font_size, 268
y_ticks_values_ioflags, 268
y_ticks_values_precision, 268
y_values_color, 269
y_values_font_size, 269
y_values_ioflags, 269
y_values_on, 269
y_values_precision, 270
y_values_rotation, 270
y_value_ioflags, 268
y_value_precision, 268
svg_2d_plot_series
Header < boost/svg_plot/svg_2d_plot.hpp >, 275
svg_2d_plot_series public member functions
bar_opt, 276
bar_width, 276
bezier_on, 277
limits_count, 277
line_on, 277
line_style, 277
line_width, 278
shape, 278
size, 278
values_count, 278
svg_boxplot
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_boxplot protected member functions
clear_background, 316
clear_grids, 316
clear_legend, 316
clear_plot_background, 316
clear_points, 316
clear_title, 316
clear_x_axis, 316
clear_y_axis, 316
draw_legend, 316

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

place_legend_box, 317
size_legend_box, 317
svg_boxplot public member functions
autoscale, 288
autoscale_check_limits, 288
autoscale_plusminus, 288
axes_on, 288
axis_color, 288
axis_width, 289
background_border_color, 289
background_border_width, 289
background_color, 289
boost_license_on, 289
box_border, 289
box_fill, 290
box_width, 290
calculate_plot_window, 290
clear_all, 290
coord_precision, 290
copyright_date, 290
copyright_holder, 290
data_lines_width, 291
document_title, 291
draw_title, 292
draw_x_axis, 292
draw_x_axis_label, 292
draw_y_axis, 292
draw_y_axis_label, 292
extreme_outlier_color, 293
extreme_outlier_fill, 293
extreme_outlier_shape, 293
extreme_outlier_size, 293
extreme_outlier_values_on, 293
image_border_margin, 294
image_border_width, 294
image_x_size, 294
image_y_size, 294
legend_background_color, 294
legend_border_color, 294
legend_box_fill_on, 294
legend_color, 295
legend_header_font_size, 295
legend_lines, 295
legend_on, 295
legend_outside, 295
legend_place, 295
legend_title, 295
legend_title_font_size, 296
legend_width, 296
license_attribution, 296
license_commercialuse, 296
license_distribution, 296
license_on, 296
license_reproduction, 296
limit_color, 296
limit_fill_color, 297
median_color, 297
median_values_on, 297

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

median_width, 297
one_sd_color, 297
outlier_color, 297
outlier_fill, 298
outlier_shape, 298
outlier_size, 298
outlier_values_on, 298
plot_background_color, 299
plot_border_color, 299
plot_border_width, 299
plot_window_on, 299
plot_window_x_left, 300
plot_window_x_right, 300
plot_window_y_bottom, 300
plot_window_y_top, 300
quartile_definition, 300
three_sd_color, 300
title, 300
title_color, 301
title_font_alignment, 301
title_font_rotation, 301
title_font_size, 301
title_on, 302
two_sd_color, 302
update_image, 302
whisker_length, 303
x_autoscale, 303
x_auto_max_value, 303
x_auto_min_value, 303
x_auto_ticks, 303
x_auto_tick_interval, 303
x_axis_color, 304
x_axis_label_color, 304
x_axis_on, 304
x_axis_position, 304
x_axis_width, 304
x_df_color, 304
x_df_on, 304
x_label, 305
x_label_color, 305
x_label_font_size, 305
x_label_on, 305
x_label_text, 306
x_label_units, 306
x_label_units_on, 306
x_label_width, 306
x_major_grid_color, 306
x_major_grid_on, 306
x_major_grid_width, 306
x_major_interval, 306
x_major_labels, 307
x_major_labels_side, 307
x_major_label_rotation, 307
x_major_tick, 307
x_major_tick_color, 307
x_major_tick_length, 307
x_major_tick_width, 307
x_max, 307

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

x_min, 308
x_minor_grid_color, 308
x_minor_grid_on, 308
x_minor_grid_width, 308
x_minor_interval, 308
x_minor_tick_color, 308
x_minor_tick_length, 308
x_minor_tick_width, 309
x_min_ticks, 308
x_num_minor_ticks, 309
x_plusminus_color, 309
x_plusminus_on, 309
x_prefix, 309
x_separator, 309
x_steps, 309
x_suffix, 309
x_ticks_down_on, 310
x_ticks_on_window_or_axis, 310
x_ticks_up_on, 310
x_ticks_values_color, 310
x_ticks_values_font_size, 311
x_ticks_values_ioflags, 311
x_ticks_values_precision, 311
x_tick_color, 310
x_tick_length, 310
x_tight, 311
x_values_color, 312
x_values_font_size, 312
x_values_ioflags, 312
x_values_on, 312
x_values_precision, 312
x_values_rotation, 312
x_value_font_size, 311
x_value_ioflags, 311
x_value_precision, 311
x_with_zero, 312
y_autoscale, 313
y_axis_color, 313
y_axis_on, 313
y_axis_position, 313
y_label, 314
y_labels_strip_e0s, 314
y_label_color, 314
y_label_text, 314
y_label_units, 314
y_minor_interval, 315
svg_boxplot_series
Header < boost/svg_plot/svg_boxplot.hpp >, 319
svg_boxplot_series public member functions
axis_color, 320
axis_width, 321
box_border, 321
box_fill, 321
box_width, 322
calculate_quantiles, 322
extreme_outlier_color, 322
extreme_outlier_fill, 323
extreme_outlier_shape, 323

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

extreme_outlier_size, 323
max_whisker_color, 323
max_whisker_width, 323
median_color, 324
median_width, 324
min_whisker_color, 324
min_whisker_width, 325
outlier_color, 325
outlier_fill, 325
outlier_shape, 325
outlier_size, 326
quartile_definition, 326
title, 326
whisker_length, 327
svg_color
Header < boost/svg_plot/svg_color.hpp >, 328
svg_color public construct/copy/destruct
my_blank, 328
svg_element
Header < boost/svg_plot/detail/svg_tag.hpp >, 113, 116, 119, 122, 129, 133, 144, 148, 151, 155, 159, 166
svg_element public member functions
class_id, 156
clip_id, 156
id, 156
svg_style
Header < boost/svg_plot/svg_style.hpp >, 352
symbols
Header < boost/svg_plot/svg_1d_plot.hpp >, 226
svg_1d_plot_series public member functions, 229

T

t
Description, 178
text
Header < boost/svg_plot/detail/svg_tag.hpp >, 159, 166
text_element public member functions, 162
tspan_element public member functions, 170
text_element
Header < boost/svg_plot/detail/svg_tag.hpp >, 159
text_element public member functions
alignment, 160
class_id, 160
clip_id, 161
id, 161
text, 162
text_element_text
Header < boost/svg_plot/detail/svg_tag.hpp >, 164
text_length
Header < boost/svg_plot/detail/svg_tag.hpp >, 166
tspan_element public member functions, 170
text_parent
Header < boost/svg_plot/detail/svg_tag.hpp >, 164, 165, 166
text_style
Header < boost/svg_plot/svg_style.hpp >, 355
three_sd_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 212
svg_2d_plot public member functions, 247
svg_boxplot public member functions, 300
ticks_labels_style
Header < boost/svg_plot/svg_style.hpp >, 358
ticks_labels_style public member functions
label_length, 359
longest_label, 359
title
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280, 319
svg_1d_plot public member functions, 212
svg_2d_plot public member functions, 247
svg_boxplot public member functions, 300
svg_boxplot_series public member functions, 326
title_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 212
svg_2d_plot public member functions, 248
svg_boxplot public member functions, 301
title_font_alignment
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 212
svg_2d_plot public member functions, 248
svg_boxplot public member functions, 301
title_font_rotation
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 213
svg_2d_plot public member functions, 248
svg_boxplot public member functions, 301
title_font_size
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 213
svg_2d_plot public member functions, 248
svg_boxplot public member functions, 301
title_on
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 213
svg_2d_plot public member functions, 249
svg_boxplot public member functions, 302
ts
Description, 367
tspan_element
Header < boost/svg_plot/detail/svg_tag.hpp >, 166
tspan_element public member functions
class_id, 167

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

clip_id, 167
dx, 168
dy, 168
fill_color, 168
fill_on, 168
font_size, 168
id, 169
rotation, 169
text, 170
text_length, 170
use_style, 170
x, 171
y, 171
Tutorial: 2D Special Features
show_1d_plot_settings, 59
two_sd_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 213
svg_2d_plot public member functions, 249
svg_boxplot public member functions, 302
types
Header < boost/svg_plot/uncertain.hpp >, 371
unc public member functions, 372
t_or_b
Header < boost/svg_plot/show_1d_settings.hpp >, 182

U

unc
Header < boost/svg_plot/uncertain.hpp >, 371
unc public member functions
deg_free, 372
types, 372
uncertainty, 372
value, 372
uncertainty
Header < boost/svg_plot/uncertain.hpp >, 371
unc public member functions, 372
unc_of
Header < boost/svg_plot/uncertain.hpp >, 370, 376, 377
update_image
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 214
svg_2d_plot private member functions, 271
svg_boxplot public member functions, 302
use_style
Header < boost/svg_plot/detail/svg_tag.hpp >, 166
tspan_element public member functions, 170

V

value
Header < boost/svg_plot/uncertain.hpp >, 371
unc public member functions, 372
values_count

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Header < boost/svg_plot/svg_2d_plot.hpp >, 275
svg_2d_plot_series public member functions, 278
value_of
Header < boost/svg_plot/uncertain.hpp >, 370, 380, 381
value_style
Header < boost/svg_plot/svg_style.hpp >, 361

W

whisker_length
Header < boost/svg_plot/svg_boxplot.hpp >, 280, 319
svg_boxplot public member functions, 303
svg_boxplot_series public member functions, 327
width
axis_line_style public member functions, 341
bar_style public member functions, 343
box_style public member functions, 345
Demonstration of using 1D data that includes information about its Uncertainty, 52
Header < boost/svg_plot/svg_style.hpp >, 340, 342, 344, 347
plot_line_style public member functions, 348
width_on
Header < boost/svg_plot/svg_fwd.hpp >, 337

X

x
Header < boost/svg_plot/detail/svg_tag.hpp >, 166
tspan_element public member functions, 171
xy_autoscale
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 258
xy_values_on
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 258
x_autoscale
1-D Auto scaling Examples, 29
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 214
svg_2d_plot public member functions, 250
svg_boxplot public member functions, 303
x_auto_max_value
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 214
svg_2d_plot public member functions, 249
svg_boxplot public member functions, 303
x_auto_min_value
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 214
svg_2d_plot public member functions, 249
svg_boxplot public member functions, 303
x_auto_ticks
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 214
svg_2d_plot public member functions, 249
svg_boxplot public member functions, 303
x_auto_tick_interval
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 214
svg_2d_plot public member functions, 249
svg_boxplot public member functions, 303
x_axis_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 214
svg_2d_plot public member functions, 250
svg_boxplot public member functions, 304
x_axis_label_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 215
svg_2d_plot public member functions, 250
svg_boxplot public member functions, 304
x_axis_on
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 215
svg_2d_plot public member functions, 250
svg_boxplot public member functions, 304
x_axis_position
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 215
svg_2d_plot public member functions, 250
svg_boxplot public member functions, 304
x_axis_width
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 215
svg_2d_plot public member functions, 250
svg_boxplot public member functions, 304
x_df_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 215
svg_2d_plot public member functions, 251
svg_boxplot public member functions, 304
x_df_on
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 215

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

svg_2d_plot public member functions, 251
svg_boxplot public member functions, 304
x_label
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 215
svg_2d_plot public member functions, 251
svg_boxplot public member functions, 305
x_label_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 215
svg_2d_plot public member functions, 251
svg_boxplot public member functions, 305
x_label_font_size
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 216
svg_2d_plot public member functions, 251
svg_boxplot public member functions, 305
x_label_on
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 216
svg_2d_plot public member functions, 251
svg_boxplot public member functions, 305
x_label_text
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_boxplot public member functions, 306
x_label_units
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 216
svg_2d_plot public member functions, 251
svg_boxplot public member functions, 306
x_label_units_on
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 216
svg_2d_plot public member functions, 252
svg_boxplot public member functions, 306
x_label_width
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 216
svg_2d_plot public member functions, 252
svg_boxplot public member functions, 306
x_major_grid_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

svg_1d_plot public member functions, 216
svg_2d_plot public member functions, 252
svg_boxplot public member functions, 306
x_major_grid_on
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 217
svg_2d_plot public member functions, 252
svg_boxplot public member functions, 306
x_major_grid_width
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 217
svg_2d_plot public member functions, 252
svg_boxplot public member functions, 306
x_major_interval
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 217
svg_2d_plot public member functions, 252
svg_boxplot public member functions, 306
x_major_labels
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_boxplot public member functions, 307
x_major_labels_side
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 217
svg_2d_plot public member functions, 253
svg_boxplot public member functions, 307
x_major_label_rotation
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 217
svg_2d_plot public member functions, 252
svg_boxplot public member functions, 307
x_major_tick
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 217
svg_2d_plot public member functions, 253
svg_boxplot public member functions, 307
x_major_tick_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 217
svg_2d_plot public member functions, 253
svg_boxplot public member functions, 307
x_major_tick_length
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 217
svg_2d_plot public member functions, 253
svg_boxplot public member functions, 307
x_major_tick_width
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 218
svg_2d_plot public member functions, 253
svg_boxplot public member functions, 307
x_max
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 218
svg_2d_plot public member functions, 253
svg_boxplot public member functions, 307
x_min
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 218
svg_2d_plot public member functions, 253
svg_boxplot public member functions, 308
x_minor_grid_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 218
svg_2d_plot public member functions, 254
svg_boxplot public member functions, 308
x_minor_grid_on
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 218
svg_2d_plot public member functions, 254
svg_boxplot public member functions, 308
x_minor_grid_width
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 218
svg_2d_plot public member functions, 254
svg_boxplot public member functions, 308
x_minor_interval
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 218
svg_2d_plot public member functions, 254
svg_boxplot public member functions, 308
x_minor_tick_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 219

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

svg_2d_plot public member functions, 254
svg_boxplot public member functions, 308
x_minor_tick_length
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 219
svg_2d_plot public member functions, 254
svg_boxplot public member functions, 308
x_minor_tick_width
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 219
svg_2d_plot public member functions, 254
svg_boxplot public member functions, 309
x_min_ticks
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 218
svg_2d_plot public member functions, 253
svg_boxplot public member functions, 308
x_num_minor_ticks
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 219
svg_2d_plot public member functions, 254
svg_boxplot public member functions, 309
x_plusminus_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 219
svg_2d_plot public member functions, 255
svg_boxplot public member functions, 309
x_plusminus_on
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 219
svg_2d_plot public member functions, 255
svg_boxplot public member functions, 309
x_prefix
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 219
svg_2d_plot public member functions, 255
svg_boxplot public member functions, 309
x_separator
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 219
svg_2d_plot public member functions, 255
svg_boxplot public member functions, 309

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

x_size
Header < boost/svg_plot/svg.hpp >, 190
Header < boost/svg_plot/svg_fwd.hpp >, 337
svg public member functions, 196
x_steps
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 220
svg_2d_plot public member functions, 255
svg_boxplot public member functions, 309
x_suffix
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 220
svg_2d_plot public member functions, 255
svg_boxplot public member functions, 309
x_ticks_down_on
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 220
svg_2d_plot public member functions, 255
svg_boxplot public member functions, 310
x_ticks_on_window_or_axis
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 220
svg_2d_plot public member functions, 255
svg_boxplot public member functions, 310
x_ticks_up_on
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 220
svg_2d_plot public member functions, 256
svg_boxplot public member functions, 310
x_ticks_values_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 220
svg_2d_plot public member functions, 256
svg_boxplot public member functions, 310
x_ticks_values_font_size
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 220
svg_2d_plot public member functions, 256
svg_boxplot public member functions, 311
x_ticks_values_ioflags
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 220

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

svg_2d_plot public member functions, 256
svg_boxplot public member functions, 311
x_ticks_values_precision
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 221
svg_2d_plot public member functions, 256
svg_boxplot public member functions, 311
x_tick_color
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_boxplot public member functions, 310
x_tick_length
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_boxplot public member functions, 310
x_tight
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 221
svg_2d_plot public member functions, 256
svg_boxplot public member functions, 311
x_values_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 221
svg_2d_plot public member functions, 257
svg_boxplot public member functions, 312
x_values_font_size
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 221
svg_2d_plot public member functions, 257
svg_boxplot public member functions, 312
x_values_ioflags
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 222
svg_2d_plot public member functions, 257
svg_boxplot public member functions, 312
x_values_on
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 222
svg_2d_plot public member functions, 257
svg_boxplot public member functions, 312
x_values_precision
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 222
svg_2d_plot public member functions, 257
svg_boxplot public member functions, 312
x_values_rotation

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 222
svg_2d_plot public member functions, 258
svg_boxplot public member functions, 312
x_value_font_size
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 221
svg_2d_plot public member functions, 256
svg_boxplot public member functions, 311
x_value_ioflags
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 221
svg_2d_plot public member functions, 257
svg_boxplot public member functions, 311
x_value_precision
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 221
svg_2d_plot public member functions, 257
svg_boxplot public member functions, 311
x_with_zero
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 222
svg_2d_plot public member functions, 258
svg_boxplot public member functions, 312

Y

y
Header < boost/svg_plot/detail/svg_tag.hpp >, 166
tspan_element public member functions, 171
y_autoscale
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_2d_plot public member functions, 259
svg_boxplot public member functions, 313
y_axis_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 222
svg_2d_plot public member functions, 259
svg_boxplot public member functions, 313
y_axis_label_color
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 259
y_axis_on
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 222
svg_2d_plot public member functions, 259
svg_boxplot public member functions, 313
y_axis_position
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_2d_plot public member functions, 259
svg_boxplot public member functions, 313
y_axis_value_color
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 259
y_axis_width
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 260
y_df_color
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 260
y_df_on
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 260
y_label
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 222
svg_2d_plot public member functions, 260
svg_boxplot public member functions, 314
y_labels_strip_e0s
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 223
svg_2d_plot public member functions, 262
svg_boxplot public member functions, 314
y_label_axis
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 260
y_label_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 223
svg_2d_plot public member functions, 261
svg_boxplot public member functions, 314
y_label_font_size
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 261
y_label_on
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 261
y_label_text
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_boxplot public member functions, 314
y_label_units
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

svg_1d_plot public member functions, 223
svg_2d_plot public member functions, 261
svg_boxplot public member functions, 314
y_label_units_on
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 261
y_label_width
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 262
y_major_grid_color
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 262
y_major_grid_on
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 262
y_major_grid_width
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 263
y_major_interval
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 263
y_major_labels_side
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 263
y_major_label_rotation
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 263
y_major_tick_color
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 264
y_major_tick_length
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 264
y_major_tick_width
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 264
y_max
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 264
y_min
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 264
y_minor_grid_color
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 264
y_minor_grid_on
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 264
y_minor_grid_width
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 265
y_minor_interval
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 223
svg_2d_plot public member functions, 265
svg_boxplot public member functions, 315

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

y_minor_tick_color
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 265
y_minor_tick_length
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 265
y_minor_tick_width
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 265
y_num_minor_ticks
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 265
y_plusminus_color
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 266
y_plusminus_on
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 266
y_prefix
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 266
y_separator
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 266
y_size
Header < boost/svg_plot/svg.hpp >, 190
Header < boost/svg_plot/svg_fwd.hpp >, 337
svg public member functions, 196
y_suffix
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 266
y_ticks_left_on
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 267
y_ticks_on_window_or_axis
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 267
y_ticks_right_on
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 267
y_ticks_values_color
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 267
y_ticks_values_font_size
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 268
y_ticks_values_ioflags
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 268
y_ticks_values_precision
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 268
y_values_color
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 269
y_values_font_size
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 269

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

y_values_ioflags
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 269
y_values_on
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 269
y_values_precision
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 270
y_values_rotation
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 270
y_value_ioflags
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 268
y_value_precision
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 268

Index

Symbols

1-D Auto scaling Examples
scale_axis, 29
show_1d_plot_settings, 29
x_autoscale, 29
1-D Data Values Examples
show_1d_plot_settings, 41

A

alignment
Header < boost/svg_plot/detail/svg_tag.hpp >, 159
text_element public member functions, 160
author
Header < boost/svg_plot/svg.hpp >, 190
Header < boost/svg_plot/svg_fwd.hpp >, 337
Implementation & Rationale, 384
svg public member functions, 192
autoscale
Demonstration of using 1D data that includes information about its uncertainty, 74
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 205
svg_2d_plot public member functions, 240
svg_boxplot public member functions, 288
autoscale_check_limits
Demonstration of using 1D data that includes information about its uncertainty, 74
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 205
svg_2d_plot public member functions, 240
svg_boxplot public member functions, 288
autoscale_plusminus
Demonstration of using 1D data that includes information about its uncertainty, 74

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 205
svg_2d_plot public member functions, 241
svg_boxplot public member functions, 288
axes_on
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 205
svg_2d_plot public member functions, 241
svg_boxplot public member functions, 288
axis_color
Header < boost/svg_plot/svg_boxplot.hpp >, 280, 319
svg_boxplot public member functions, 288
svg_boxplot_series public member functions, 320
axis_line_style
Header < boost/svg_plot/svg_style.hpp >, 340
axis_line_style public member functions
color, 341
position, 341
width, 341
axis_width
Header < boost/svg_plot/svg_boxplot.hpp >, 280, 319
svg_boxplot public member functions, 289
svg_boxplot_series public member functions, 321

B

background_border_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 205
svg_2d_plot public member functions, 241
svg_boxplot public member functions, 289
background_border_width
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 205
svg_2d_plot public member functions, 241
svg_boxplot public member functions, 289
background_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 205
svg_2d_plot public member functions, 241
svg_boxplot public member functions, 289
bar_opt
bar_style public member functions, 342
Header < boost/svg_plot/svg_2d_plot.hpp >, 275
Header < boost/svg_plot/svg_style.hpp >, 342
svg_2d_plot_series public member functions, 276
bar_style
Header < boost/svg_plot/svg_style.hpp >, 342

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

bar_style public member functions
bar_opt, 342
width, 343
bar_width
Header < boost/svg_plot/svg_2d_plot.hpp >, 275
svg_2d_plot_series public member functions, 276
bezier_on
Header < boost/svg_plot/svg_1d_plot.hpp >, 226
Header < boost/svg_plot/svg_2d_plot.hpp >, 275
svg_1d_plot_series public member functions, 227
svg_2d_plot_series public member functions, 277
boost_license
Header < boost/svg_plot/svg_fwd.hpp >, 337
boost_license_on
Header < boost/svg_plot/svg.hpp >, 190
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg public member functions, 192
svg_1d_plot public member functions, 205
svg_2d_plot public member functions, 241
svg_boxplot public member functions, 289
box_border
Header < boost/svg_plot/svg_boxplot.hpp >, 280, 319
svg_boxplot public member functions, 289
svg_boxplot_series public member functions, 321
box_fill
Header < boost/svg_plot/svg_boxplot.hpp >, 280, 319
svg_boxplot public member functions, 290
svg_boxplot_series public member functions, 321
box_style
Header < boost/svg_plot/svg_style.hpp >, 344
box_style public member functions
fill, 345
margin, 345
stroke, 345
width, 345
box_width
Header < boost/svg_plot/svg_boxplot.hpp >, 280, 319
svg_boxplot public member functions, 290
svg_boxplot_series public member functions, 322

C

calculate_plot_window
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 206
svg_2d_plot private member functions, 270
svg_boxplot public member functions, 290
calculate_quantiles
Header < boost/svg_plot/svg_boxplot.hpp >, 319
svg_boxplot_series public member functions, 322
calculate_transform
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
svg_1d_plot public member functions, 206
circle_element

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Header < boost/svg_plot/detail/svg_tag.hpp >, 113
circle_element public member functions
class_id, 114
clip_id, 114
id, 114
class_id
circle_element public member functions, 114
clip_path_element public member functions, 116
ellipse_element public member functions, 120
g_element public member functions, 123
Header < boost/svg_plot/detail/svg_tag.hpp >, 113, 116, 119, 122, 129, 133, 140, 144, 148, 151, 155, 159, 166
line_element public member functions, 130
path_element public member functions, 134
polygon_element public member functions, 141
polyline_element public member functions, 145
qurve_element public member functions, 149
rect_element public member functions, 152
svg_element public member functions, 156
text_element public member functions, 160
tspan_element public member functions, 167
clear
g_element public member functions, 123
Header < boost/svg_plot/detail/svg_tag.hpp >, 122
clear_all
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot protected member functions, 223
svg_2d_plot protected member functions, 271
svg_boxplot public member functions, 290
clear_background
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot protected member functions, 223
svg_2d_plot protected member functions, 272
svg_boxplot protected member functions, 316
clear_grids
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot protected member functions, 223
svg_2d_plot protected member functions, 272
svg_boxplot protected member functions, 316
clear_legend
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot protected member functions, 223
svg_2d_plot protected member functions, 272
svg_boxplot protected member functions, 316
clear_plot_background
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot protected member functions, 223
svg_2d_plot protected member functions, 272
svg_boxplot protected member functions, 316

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

clear_points
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot protected member functions, 224
svg_2d_plot protected member functions, 272
svg_boxplot protected member functions, 316
clear_title
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot protected member functions, 224
svg_2d_plot protected member functions, 272
svg_boxplot protected member functions, 316
clear_x_axis
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot protected member functions, 224
svg_2d_plot protected member functions, 272
svg_boxplot protected member functions, 316
clear_y_axis
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot protected member functions, 224
svg_2d_plot protected member functions, 272
svg_boxplot protected member functions, 316
clip_id
circle_element public member functions, 114
clip_path_element public member functions, 117
ellipse_element public member functions, 120
g_element public member functions, 123
Header < boost/svg_plot/detail/svg_tag.hpp >, 113, 116, 119, 122, 129, 133, 140, 144, 148, 151, 155, 159, 166
line_element public member functions, 130
path_element public member functions, 134
polygon_element public member functions, 141
polyline_element public member functions, 145
qurve_element public member functions, 149
rect_element public member functions, 152
svg_element public member functions, 156
text_element public member functions, 161
tspan_element public member functions, 167
clip_path_element
Header < boost/svg_plot/detail/svg_tag.hpp >, 116
clip_path_element public member functions
class_id, 116
clip_id, 117
id, 117
close_to
Header < boost/svg_plot/detail/FP_compare.hpp >, 106
close_to public member functions
size, 106
strength, 106
color
axis_line_style public member functions, 341
Header < boost/svg_plot/svg_style.hpp >, 340
constant_to_rgb

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Header < boost/svg_plot/svg_color.hpp >, 327, 332
coord_precision
Header < boost/svg_plot/svg.hpp >, 190
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg public member functions, 193
svg_1d_plot public member functions, 206
svg_2d_plot public member functions, 241
svg_boxplot public member functions, 290
coord_precison
Header < boost/svg_plot/svg_fwd.hpp >, 337
copyright_date
Header < boost/svg_plot/svg.hpp >, 190
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
Header < boost/svg_plot/svg_fwd.hpp >, 337
svg public member functions, 193
svg_1d_plot public member functions, 206
svg_2d_plot public member functions, 241
svg_boxplot public member functions, 290
copyright_holder
Header < boost/svg_plot/svg.hpp >, 190
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
Header < boost/svg_plot/svg_fwd.hpp >, 337
svg public member functions, 193
svg_1d_plot public member functions, 206
svg_2d_plot public member functions, 241
svg_boxplot public member functions, 290

D

data_lines_width
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 206
svg_2d_plot public member functions, 242
svg_boxplot public member functions, 291
default_plot_point_style
Header < boost/svg_plot/svg_style.hpp >, 338
deg_free
Header < boost/svg_plot/uncertain.hpp >, 371
unc public member functions, 372
Demonstration of using 1D data that includes information about its Uncertainty
width, 52
Demonstration of using 1D data that includes information about its uncertainty
autoscale, 74
autoscale_check_limits, 74
autoscale_plusminus, 74
Description
my_color, 336
p, 176
p0, 177
r, 179

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

t, 178
ts, 367
description
Header < boost/svg_plot/svg.hpp >, 190
svg public member functions, 193
document_size
Header < boost/svg_plot/svg.hpp >, 190
svg public member functions, 194
document_title
Header < boost/svg_plot/svg.hpp >, 190
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
Header < boost/svg_plot/svg_fwd.hpp >, 337
svg public member functions, 194
svg_1d_plot public member functions, 206
svg_2d_plot public member functions, 242
svg_boxplot public member functions, 291
draw_axes
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
svg_1d_plot public member functions, 206
draw_bars
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot private member functions, 270
draw_histogram
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot private member functions, 270
draw_legend
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot protected member functions, 224
svg_2d_plot protected member functions, 272
svg_boxplot protected member functions, 316
draw_plot_lines
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot private member functions, 271
draw_plot_points
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot private member functions, 271
draw_title
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot protected member functions, 224
svg_2d_plot protected member functions, 273
svg_boxplot public member functions, 292
draw_x_axis
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot protected member functions, 225
svg_2d_plot protected member functions, 273
svg_boxplot public member functions, 292
draw_x_axis_label
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

svg_1d_plot protected member functions, 225
svg_2d_plot protected member functions, 273
svg_boxplot public member functions, 292
draw_y_axis
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_2d_plot private member functions, 271
svg_boxplot public member functions, 292
draw_y_axis_label
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_2d_plot private member functions, 271
svg_boxplot public member functions, 292
dx
Header < boost/svg_plot/detail/svg_tag.hpp >, 166
tspan_element public member functions, 168
dy
Header < boost/svg_plot/detail/svg_tag.hpp >, 166
tspan_element public member functions, 168

E

ellipse_element
Header < boost/svg_plot/detail/svg_tag.hpp >, 119
ellipse_element public member functions
class_id, 120
clip_id, 120
id, 120
epsilon
Header < boost/svg_plot/detail/FP_compare.hpp >, 105
extreme_outlier_color
Header < boost/svg_plot/svg_boxplot.hpp >, 280, 319
svg_boxplot public member functions, 293
svg_boxplot_series public member functions, 322
extreme_outlier_fill
Header < boost/svg_plot/svg_boxplot.hpp >, 280, 319
svg_boxplot public member functions, 293
svg_boxplot_series public member functions, 323
extreme_outlier_shape
Header < boost/svg_plot/svg_boxplot.hpp >, 280, 319
svg_boxplot public member functions, 293
svg_boxplot_series public member functions, 323
extreme_outlier_size
Header < boost/svg_plot/svg_boxplot.hpp >, 280, 319
svg_boxplot public member functions, 293
svg_boxplot_series public member functions, 323
extreme_outlier_values_on
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_boxplot public member functions, 293

F

fill
box_style public member functions, 345
Header < boost/svg_plot/svg_style.hpp >, 344
fill_color
Header < boost/svg_plot/detail/svg_tag.hpp >, 166
Header < boost/svg_plot/svg_1d_plot.hpp >, 226
Header < boost/svg_plot/svg_fwd.hpp >, 337

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

svg_1d_plot_series public member functions, 227
tspan_element public member functions, 168
fill_on
Header < boost/svg_plot/detail/svg_tag.hpp >, 133, 166
Header < boost/svg_plot/svg_fwd.hpp >, 337
path_element public member functions, 134
tspan_element public member functions, 168
font_size
Header < boost/svg_plot/detail/svg_tag.hpp >, 166
tspan_element public member functions, 168
fpt_abs
Header < boost/svg_plot/detail/FP_compare.hpp >, 105

G

g_element
Header < boost/svg_plot/detail/svg_tag.hpp >, 122
g_element public member functions
class_id, 123
clear, 123
clip_id, 123
id, 124
push_back, 125
size, 126

H

Header < boost/svg_plot/detail/auto_axes.hpp >
isfinite, 87, 89
mnmx, 87, 90
rounddown10, 87, 93
rounddown2, 87, 94
rounddown5, 87
roundup10, 87, 95
roundup2, 87, 96
roundup5, 87, 97
scale_axis, 87, 100
show, 87, 101
Header < boost/svg_plot/detail/FP_compare.hpp >
close_to, 106
epsilon, 105
fpt_abs, 105
max_value, 105
min_value, 105
safe_fpt_division, 105
size, 106, 107
smallest, 107
strength, 106
Header < boost/svg_plot/detail/svg_tag.hpp >
alignment, 159
circle_element, 113
class_id, 113, 116, 119, 122, 129, 133, 140, 144, 148, 151, 155, 159, 166
clear, 122
clip_id, 113, 116, 119, 122, 129, 133, 140, 144, 148, 151, 155, 159, 166
clip_path_element, 116
dx, 166
dy, 166
ellipse_element, 119
fill_color, 166

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

fill_on, 133, 166
font_size, 166
g_element, 122
id, 113, 116, 119, 122, 129, 133, 140, 144, 148, 151, 155, 159, 166
line_element, 129
path_element, 133
polyline_element, 144
push_back, 122
qurve_element, 148
rect_element, 151
rotation, 166
size, 122
svg, 113, 116, 119, 122, 129, 133, 144, 148, 151, 159, 164, 166
svg_element, 113, 116, 119, 122, 129, 133, 144, 148, 151, 155, 159, 166
text, 159, 166
text_element, 159
text_element_text, 164
text_length, 166
text_parent, 164, 165, 166
tspan_element, 166
use_style, 166
x, 166
y, 166
Header < boost/svg_plot/show_1d_settings.hpp >
l_or_r, 182
t_or_b, 182
Header < boost/svg_plot/svg.hpp >
author, 190
boost_license_on, 190
coord_precision, 190
copyright_date, 190
copyright_holder, 190
description, 190
document_size, 190
document_title, 190
image_filename, 190
license, 190
license_on, 190
size, 190
svg, 190
x_size, 190
y_size, 190
Header < boost/svg_plot/svg_1d_plot.hpp >
autoscale, 198
autoscale_check_limits, 198
autoscale_plusminus, 198
axes_on, 198
background_border_color, 198
background_border_width, 198
background_color, 198
bezier_on, 226
boost_license_on, 198
calculate_plot_window, 198
calculate_transform, 198
clear_all, 198
clear_background, 198
clear_grids, 198
clear_legend, 198

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

clear_plot_background, 198
clear_points, 198
clear_title, 198
clear_x_axis, 198
clear_y_axis, 198
coord_precision, 198
copyright_date, 198
copyright_holder, 198
data_lines_width, 198
document_title, 198
draw_axes, 198
draw_legend, 198
draw_title, 198
draw_x_axis, 198
draw_x_axis_label, 198
fill_color, 226
image_border_margin, 198
image_border_width, 198
image_x_size, 198
image_y_size, 198
legend_background_color, 198
legend_border_color, 198
legend_box_fill_on, 198
legend_color, 198
legend_header_font_size, 198
legend_lines, 198
legend_on, 198
legend_outside, 198
legend_place, 198
legend_title, 198
legend_title_font_size, 198
legend_width, 198
license_attribution, 198
license_commercialuse, 198
license_distribution, 198
license_on, 198
license_reproduction, 198
limit_color, 198
limit_fill_color, 198
line_on, 226
line_width, 226
one_sd_color, 198
place_legend_box, 198
plot_background_color, 198
plot_border_color, 198
plot_border_width, 198
plot_window_on, 198
plot_window_x_left, 198
plot_window_x_right, 198
plot_window_y_bottom, 198
plot_window_y_top, 198
series_count, 226
series_limits_count, 226
set_ids, 198
shape, 226
size, 226
size_legend_box, 198
strip_e0s, 197, 230

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

stroke_color, 226
svg_1d_plot, 198
svg_1d_plot_series, 226
symbols, 226
three_sd_color, 198
title, 198
title_color, 198
title_font_alignment, 198
title_font_rotation, 198
title_font_size, 198
title_on, 198
two_sd_color, 198
update_image, 198
x_autoscale, 198
x_auto_max_value, 198
x_auto_min_value, 198
x_auto_ticks, 198
x_auto_tick_interval, 198
x_axis_color, 198
x_axis_label_color, 198
x_axis_on, 198
x_axis_position, 198
x_axis_width, 198
x_df_color, 198
x_df_on, 198
x_label, 198
x_label_color, 198
x_label_font_size, 198
x_label_on, 198
x_label_units, 198
x_label_units_on, 198
x_label_width, 198
x_major_grid_color, 198
x_major_grid_on, 198
x_major_grid_width, 198
x_major_interval, 198
x_major_labels_side, 198
x_major_label_rotation, 198
x_major_tick, 198
x_major_tick_color, 198
x_major_tick_length, 198
x_major_tick_width, 198
x_max, 198
x_min, 198
x_minor_grid_color, 198
x_minor_grid_on, 198
x_minor_grid_width, 198
x_minor_interval, 198
x_minor_tick_color, 198
x_minor_tick_length, 198
x_minor_tick_width, 198
x_min_ticks, 198
x_num_minor_ticks, 198
x_plusminus_color, 198
x_plusminus_on, 198
x_prefix, 198
x_separator, 198
x_steps, 198

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

x_suffix, 198
x_ticks_down_on, 198
x_ticks_on_window_or_axis, 198
x_ticks_up_on, 198
x_ticks_values_color, 198
x_ticks_values_font_size, 198
x_ticks_values_ioflags, 198
x_ticks_values_precision, 198
x_tight, 198
x_values_color, 198
x_values_font_size, 198
x_values_ioflags, 198
x_values_on, 198
x_values_precision, 198
x_values_rotation, 198
x_value_font_size, 198
x_value_ioflags, 198
x_value_precision, 198
x_with_zero, 198
y_axis_color, 198
y_axis_on, 198
y_label, 198
y_labels_strip_e0s, 198
y_label_color, 198
y_label_units, 198
y_minor_interval, 198
Header < boost/svg_plot/svg_2d_plot.hpp >
autoscale, 231
autoscale_check_limits, 231
autoscale_plusminus, 231
axes_on, 231
background_border_color, 231
background_border_width, 231
background_color, 231
bar_opt, 275
bar_width, 275
bezier_on, 275
boost_license_on, 231
calculate_plot_window, 231
clear_all, 231
clear_background, 231
clear_grids, 231
clear_legend, 231
clear_plot_background, 231
clear_points, 231
clear_title, 231
clear_x_axis, 231
clear_y_axis, 231
coord_precision, 231
copyright_date, 231
copyright_holder, 231
data_lines_width, 231
document_title, 231
draw_bars, 231
draw_histogram, 231
draw_legend, 231
draw_plot_lines, 231
draw_plot_points, 231

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

draw_title, 231
draw_x_axis, 231
draw_x_axis_label, 231
draw_y_axis, 231
draw_y_axis_label, 231
image_border_margin, 231
image_border_width, 231
image_x_size, 231
image_y_size, 231
legend_background_color, 231
legend_border_color, 231
legend_box_fill_on, 231
legend_color, 231
legend_header_font_size, 231
legend_lines, 231
legend_on, 231
legend_outside, 231
legend_place, 231
legend_title, 231
legend_title_font_size, 231
legend_width, 231
license_attribution, 231
license_commercialuse, 231
license_distribution, 231
license_on, 231
license_reproduction, 231
limits_count, 275
limit_color, 231
limit_fill_color, 231
line_on, 275
line_style, 275
line_width, 275
one_sd_color, 231
place_legend_box, 231
plot_background_color, 231
plot_border_color, 231
plot_border_width, 231
plot_window_on, 231
plot_window_x_left, 231
plot_window_x_right, 231
plot_window_y_bottom, 231
plot_window_y_top, 231
set_ids, 231
shape, 275
size, 275
size_legend_box, 231
svg_2d_plot, 231
svg_2d_plot_series, 275
three_sd_color, 231
title, 231
title_color, 231
title_font_alignment, 231
title_font_rotation, 231
title_font_size, 231
title_on, 231
two_sd_color, 231
update_image, 231
values_count, 275

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

xy_autoscale, 231
xy_values_on, 231
x_autoscale, 231
x_auto_max_value, 231
x_auto_min_value, 231
x_auto_ticks, 231
x_auto_tick_interval, 231
x_axis_color, 231
x_axis_label_color, 231
x_axis_on, 231
x_axis_position, 231
x_axis_width, 231
x_df_color, 231
x_df_on, 231
x_label, 231
x_label_color, 231
x_label_font_size, 231
x_label_on, 231
x_label_units, 231
x_label_units_on, 231
x_label_width, 231
x_major_grid_color, 231
x_major_grid_on, 231
x_major_grid_width, 231
x_major_interval, 231
x_major_labels_side, 231
x_major_label_rotation, 231
x_major_tick, 231
x_major_tick_color, 231
x_major_tick_length, 231
x_major_tick_width, 231
x_max, 231
x_min, 231
x_minor_grid_color, 231
x_minor_grid_on, 231
x_minor_grid_width, 231
x_minor_interval, 231
x_minor_tick_color, 231
x_minor_tick_length, 231
x_minor_tick_width, 231
x_min_ticks, 231
x_num_minor_ticks, 231
x_plusminus_color, 231
x_plusminus_on, 231
x_prefix, 231
x_separator, 231
x_steps, 231
x_suffix, 231
x_ticks_down_on, 231
x_ticks_on_window_or_axis, 231
x_ticks_up_on, 231
x_ticks_values_color, 231
x_ticks_values_font_size, 231
x_ticks_values_ioflags, 231
x_ticks_values_precision, 231
x_tight, 231
x_values_color, 231
x_values_font_size, 231

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

x_values_ioflags, 231
x_values_on, 231
x_values_precision, 231
x_values_rotation, 231
x_value_font_size, 231
x_value_ioflags, 231
x_value_precision, 231
x_with_zero, 231
y_autoscale, 231
y_axis_color, 231
y_axis_label_color, 231
y_axis_on, 231
y_axis_position, 231
y_axis_value_color, 231
y_axis_width, 231
y_df_color, 231
y_df_on, 231
y_label, 231
y_labels_strip_e0s, 231
y_label_axis, 231
y_label_color, 231
y_label_font_size, 231
y_label_on, 231
y_label_units, 231
y_label_units_on, 231
y_label_width, 231
y_major_grid_color, 231
y_major_grid_on, 231
y_major_grid_width, 231
y_major_interval, 231
y_major_labels_side, 231
y_major_label_rotation, 231
y_major_tick_color, 231
y_major_tick_length, 231
y_major_tick_width, 231
y_max, 231
y_min, 231
y_minor_grid_color, 231
y_minor_grid_on, 231
y_minor_grid_width, 231
y_minor_interval, 231
y_minor_tick_color, 231
y_minor_tick_length, 231
y_minor_tick_width, 231
y_num_minor_ticks, 231
y_plusminus_color, 231
y_plusminus_on, 231
y_prefix, 231
y_separator, 231
y_suffix, 231
y_ticks_left_on, 231
y_ticks_on_window_or_axis, 231
y_ticks_right_on, 231
y_ticks_values_color, 231
y_ticks_values_font_size, 231
y_ticks_values_ioflags, 231
y_ticks_values_precision, 231
y_values_color, 231

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

y_values_font_size, 231
y_values_ioflags, 231
y_values_on, 231
y_values_precision, 231
y_values_rotation, 231
y_value_ioflags, 231
y_value_precision, 231
Header < boost/svg_plot/svg_boxplot.hpp >
autoscale, 280
autoscale_check_limits, 280
autoscale_plusminus, 280
axes_on, 280
axis_color, 280, 319
axis_width, 280, 319
background_border_color, 280
background_border_width, 280
background_color, 280
boost_license_on, 280
box_border, 280, 319
box_fill, 280, 319
box_width, 280, 319
calculate_plot_window, 280
calculate_quantiles, 319
clear_all, 280
clear_background, 280
clear_grids, 280
clear_legend, 280
clear_plot_background, 280
clear_points, 280
clear_title, 280
clear_x_axis, 280
clear_y_axis, 280
coord_precision, 280
copyright_date, 280
copyright_holder, 280
data_lines_width, 280
document_title, 280
draw_legend, 280
draw_title, 280
draw_x_axis, 280
draw_x_axis_label, 280
draw_y_axis, 280
draw_y_axis_label, 280
extreme_outlier_color, 280, 319
extreme_outlier_fill, 280, 319
extreme_outlier_shape, 280, 319
extreme_outlier_size, 280, 319
extreme_outlier_values_on, 280
image_border_margin, 280
image_border_width, 280
image_x_size, 280
image_y_size, 280
legend_background_color, 280
legend_border_color, 280
legend_box_fill_on, 280
legend_color, 280
legend_header_font_size, 280
legend_lines, 280

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

legend_on, 280
legend_outside, 280
legend_place, 280
legend_title, 280
legend_title_font_size, 280
legend_width, 280
license_attribution, 280
license_commercialuse, 280
license_distribution, 280
license_on, 280
license_reproduction, 280
limit_color, 280
limit_fill_color, 280
max_whisker_color, 319
max_whisker_width, 319
median_color, 280, 319
median_values_on, 280
median_width, 280, 319
min_whisker_color, 319
min_whisker_width, 319
one_sd_color, 280
outlier_color, 280, 319
outlier_fill, 280, 319
outlier_shape, 280, 319
outlier_size, 280, 319
outlier_values_on, 280
place_legend_box, 280
plot_background_color, 280
plot_border_color, 280
plot_border_width, 280
plot_window_on, 280
plot_window_x_left, 280
plot_window_x_right, 280
plot_window_y_bottom, 280
plot_window_y_top, 280
quartile_definition, 280, 319
size_legend_box, 280
svg_boxplot, 280
svg_boxplot_series, 319
three_sd_color, 280
title, 280, 319
title_color, 280
title_font_alignment, 280
title_font_rotation, 280
title_font_size, 280
title_on, 280
two_sd_color, 280
update_image, 280
whisker_length, 280, 319
x_autoscale, 280
x_auto_max_value, 280
x_auto_min_value, 280
x_auto_ticks, 280
x_auto_tick_interval, 280
x_axis_color, 280
x_axis_label_color, 280
x_axis_on, 280
x_axis_position, 280

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

x_axis_width, 280
x_df_color, 280
x_df_on, 280
x_label, 280
x_label_color, 280
x_label_font_size, 280
x_label_on, 280
x_label_text, 280
x_label_units, 280
x_label_units_on, 280
x_label_width, 280
x_major_grid_color, 280
x_major_grid_on, 280
x_major_grid_width, 280
x_major_interval, 280
x_major_labels, 280
x_major_labels_side, 280
x_major_label_rotation, 280
x_major_tick, 280
x_major_tick_color, 280
x_major_tick_length, 280
x_major_tick_width, 280
x_max, 280
x_min, 280
x_minor_grid_color, 280
x_minor_grid_on, 280
x_minor_grid_width, 280
x_minor_interval, 280
x_minor_tick_color, 280
x_minor_tick_length, 280
x_minor_tick_width, 280
x_min_ticks, 280
x_num_minor_ticks, 280
x_plusminus_color, 280
x_plusminus_on, 280
x_prefix, 280
x_separator, 280
x_steps, 280
x_suffix, 280
x_ticks_down_on, 280
x_ticks_on_window_or_axis, 280
x_ticks_up_on, 280
x_ticks_values_color, 280
x_ticks_values_font_size, 280
x_ticks_values_ioflags, 280
x_ticks_values_precision, 280
x_tick_color, 280
x_tick_length, 280
x_tight, 280
x_values_color, 280
x_values_font_size, 280
x_values_ioflags, 280
x_values_on, 280
x_values_precision, 280
x_values_rotation, 280
x_value_font_size, 280
x_value_ioflags, 280
x_value_precision, 280

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

x_with_zero, 280
y_autoscale, 280
y_axis_color, 280
y_axis_on, 280
y_axis_position, 280
y_label, 280
y_labels_strip_e0s, 280
y_label_color, 280
y_label_text, 280
y_label_units, 280
y_minor_interval, 280
Header < boost/svg_plot/svg_color.hpp >
constant_to_rgb, 327, 332
svg_color, 328
Header < boost/svg_plot/svg_fwd.hpp >
author, 337
boost_license, 337
coord_precison, 337
copyright_date, 337
copyright_holder, 337
document_title, 337
fill_color, 337
fill_on, 337
id, 337
is_license, 337
stroke_color, 337
stroke_on, 337
stroke_width, 337
width_on, 337
x_size, 337
y_size, 337
Header < boost/svg_plot/svg_style.hpp >
axis_line_style, 340
bar_opt, 342
bar_style, 342
box_style, 344
color, 340
default_plot_point_style, 338
fill, 344
histogram, 346
histogram_style, 346
label_length, 358
longest_label, 358
margin, 344
plot_line_style, 347
plot_point_style, 349
position, 340
shape, 349
size, 349
stroke, 344
svg_style, 352
text_style, 355
ticks_labels_style, 358
value_style, 361
width, 340, 342, 344, 347
Header < boost/svg_plot/uncertain.hpp >
deg_free, 371
types, 371

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

unc, 371
uncertainty, 371
unc_of, 370, 376, 377
value, 371
value_of, 370, 380, 381
histogram
Header < boost/svg_plot/svg_style.hpp >, 346
histogram_style public member functions, 346
histogram_style
Header < boost/svg_plot/svg_style.hpp >, 346
histogram_style public member functions
histogram, 346

I

id
circle_element public member functions, 114
clip_path_element public member functions, 117
ellipse_element public member functions, 120
g_element public member functions, 124
Header < boost/svg_plot/detail/svg_tag.hpp >, 113, 116, 119, 122, 129, 133, 140, 144, 148, 151, 155, 159, 166
Header < boost/svg_plot/svg_fwd.hpp >, 337
line_element public member functions, 130
path_element public member functions, 135
polygon_element public member functions, 142
polyline_element public member functions, 145
qurve_element public member functions, 149
rect_element public member functions, 152
svg_element public member functions, 156
text_element public member functions, 161
tspan_element public member functions, 169
image_border_margin
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 207
svg_2d_plot public member functions, 242
svg_boxplot public member functions, 294
image_border_width
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 207
svg_2d_plot public member functions, 242
svg_boxplot public member functions, 294
image_filename
Header < boost/svg_plot/svg.hpp >, 190
svg public member functions, 194
image_x_size
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 207
svg_2d_plot public member functions, 243
svg_boxplot public member functions, 294
image_y_size
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 207
svg_2d_plot public member functions, 243
svg_boxplot public member functions, 294
Implementation & Rationale
author, 384
license_attribution, 384
license_distribution, 384
license_reproduction, 384
isfinite
Header < boost/svg_plot/detail/auto_axes.hpp >, 87, 89
is_license
Header < boost/svg_plot/svg_fwd.hpp >, 337

L

label_length
Header < boost/svg_plot/svg_style.hpp >, 358
ticks_labels_style public member functions, 359
legend_background_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 207
svg_2d_plot public member functions, 243
svg_boxplot public member functions, 294
legend_border_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 208
svg_2d_plot public member functions, 243
svg_boxplot public member functions, 294
legend_box_fill_on
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 208
svg_2d_plot public member functions, 243
svg_boxplot public member functions, 294
legend_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 208
svg_2d_plot public member functions, 243
svg_boxplot public member functions, 295
legend_header_font_size
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 208
svg_2d_plot public member functions, 244
svg_boxplot public member functions, 295
legend_lines
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

svg_1d_plot public member functions, 208
svg_2d_plot public member functions, 244
svg_boxplot public member functions, 295
legend_on
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 208
svg_2d_plot public member functions, 244
svg_boxplot public member functions, 295
legend_outside
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 209
svg_2d_plot public member functions, 244
svg_boxplot public member functions, 295
legend_place
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 209
svg_2d_plot public member functions, 244
svg_boxplot public member functions, 295
legend_title
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 209
svg_2d_plot public member functions, 244
svg_boxplot public member functions, 295
legend_title_font_size
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 209
svg_2d_plot public member functions, 244
svg_boxplot public member functions, 296
legend_width
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 209
svg_2d_plot public member functions, 245
svg_boxplot public member functions, 296
license
Header < boost/svg_plot/svg.hpp >, 190
svg public member functions, 194
license_attribution
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
Implementation & Rationale, 384
svg_1d_plot public member functions, 209
svg_2d_plot public member functions, 245
svg_boxplot public member functions, 296
license_commercialuse
Header < boost/svg_plot/svg_1d_plot.hpp >, 198

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 209
svg_2d_plot public member functions, 245
svg_boxplot public member functions, 296
license_distribution
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
Implementation & Rationale, 384
svg_1d_plot public member functions, 209
svg_2d_plot public member functions, 245
svg_boxplot public member functions, 296
license_on
Header < boost/svg_plot/svg.hpp >, 190
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg public member functions, 194, 195
svg_1d_plot public member functions, 210
svg_2d_plot public member functions, 245
svg_boxplot public member functions, 296
license_reproduction
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
Implementation & Rationale, 384
svg_1d_plot public member functions, 210
svg_2d_plot public member functions, 245
svg_boxplot public member functions, 296
limits_count
Header < boost/svg_plot/svg_2d_plot.hpp >, 275
svg_2d_plot_series public member functions, 277
limit_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 210
svg_2d_plot public member functions, 245
svg_boxplot public member functions, 296
limit_fill_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 210
svg_2d_plot public member functions, 245
svg_boxplot public member functions, 297
line_element
Header < boost/svg_plot/detail/svg_tag.hpp >, 129
line_element public member functions
class_id, 130
clip_id, 130
id, 130
line_on
Header < boost/svg_plot/svg_1d_plot.hpp >, 226
Header < boost/svg_plot/svg_2d_plot.hpp >, 275
svg_1d_plot_series public member functions, 227
svg_2d_plot_series public member functions, 277

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

line_style
Header < boost/svg_plot/svg_2d_plot.hpp >, 275
svg_2d_plot_series public member functions, 277
line_width
Header < boost/svg_plot/svg_1d_plot.hpp >, 226
Header < boost/svg_plot/svg_2d_plot.hpp >, 275
svg_1d_plot_series public member functions, 227
svg_2d_plot_series public member functions, 278
longest_label
Header < boost/svg_plot/svg_style.hpp >, 358
ticks_labels_style public member functions, 359
l_or_r
Header < boost/svg_plot/show_1d_settings.hpp >, 182

M

margin
box_style public member functions, 345
Header < boost/svg_plot/svg_style.hpp >, 344
max_value
Header < boost/svg_plot/detail/FP_compare.hpp >, 105
max_whisker_color
Header < boost/svg_plot/svg_boxplot.hpp >, 319
svg_boxplot_series public member functions, 323
max_whisker_width
Header < boost/svg_plot/svg_boxplot.hpp >, 319
svg_boxplot_series public member functions, 323
median_color
Header < boost/svg_plot/svg_boxplot.hpp >, 280, 319
svg_boxplot public member functions, 297
svg_boxplot_series public member functions, 324
median_values_on
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_boxplot public member functions, 297
median_width
Header < boost/svg_plot/svg_boxplot.hpp >, 280, 319
svg_boxplot public member functions, 297
svg_boxplot_series public member functions, 324
min_value
Header < boost/svg_plot/detail/FP_compare.hpp >, 105
min_whisker_color
Header < boost/svg_plot/svg_boxplot.hpp >, 319
svg_boxplot_series public member functions, 324
min_whisker_width
Header < boost/svg_plot/svg_boxplot.hpp >, 319
svg_boxplot_series public member functions, 325
mnmx
Header < boost/svg_plot/detail/auto_axes.hpp >, 87, 90
my_blank
svg_color public construct/copy/destruct, 328
my_color
Description, 336

O

one_sd_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

svg_1d_plot public member functions, 210
svg_2d_plot public member functions, 245
svg_boxplot public member functions, 297
outlier_color
Header < boost/svg_plot/svg_boxplot.hpp >, 280, 319
svg_boxplot public member functions, 297
svg_boxplot_series public member functions, 325
outlier_fill
Header < boost/svg_plot/svg_boxplot.hpp >, 280, 319
svg_boxplot public member functions, 298
svg_boxplot_series public member functions, 325
outlier_shape
Header < boost/svg_plot/svg_boxplot.hpp >, 280, 319
svg_boxplot public member functions, 298
svg_boxplot_series public member functions, 325
outlier_size
Header < boost/svg_plot/svg_boxplot.hpp >, 280, 319
svg_boxplot public member functions, 298
svg_boxplot_series public member functions, 326
outlier_values_on
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_boxplot public member functions, 298

P

p
Description, 176
polygon_element public member functions, 142
p0
Description, 177
path_element
Header < boost/svg_plot/detail/svg_tag.hpp >, 133
path_element public member functions
class_id, 134
clip_id, 134
fill_on, 134
id, 135
place_legend_box
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot protected member functions, 225
svg_2d_plot protected member functions, 273
svg_boxplot protected member functions, 317
plot_background_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 211
svg_2d_plot public member functions, 246
svg_boxplot public member functions, 299
plot_border_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 211
svg_2d_plot public member functions, 246
svg_boxplot public member functions, 299

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

plot_border_width
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 211
svg_2d_plot public member functions, 247
svg_boxplot public member functions, 299
plot_line_style
Header < boost/svg_plot/svg_style.hpp >, 347
plot_line_style public member functions
width, 348
plot_point_style
Header < boost/svg_plot/svg_style.hpp >, 349
plot_point_style public member functions
shape, 350
size, 350
plot_window_on
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 211
svg_2d_plot public member functions, 247
svg_boxplot public member functions, 299
plot_window_x_left
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 211
svg_2d_plot public member functions, 247
svg_boxplot public member functions, 300
plot_window_x_right
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 211
svg_2d_plot public member functions, 247
svg_boxplot public member functions, 300
plot_window_y_bottom
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 212
svg_2d_plot public member functions, 247
svg_boxplot public member functions, 300
plot_window_y_top
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 212
svg_2d_plot public member functions, 247
svg_boxplot public member functions, 300
polygon_element public member functions
class_id, 141
clip_id, 141
id, 142
p, 142
polyline_element
Header < boost/svg_plot/detail/svg_tag.hpp >, 144

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

polyline_element public member functions
class_id, 145
clip_id, 145
id, 145
position
axis_line_style public member functions, 341
Header < boost/svg_plot/svg_style.hpp >, 340
push_back
g_element public member functions, 125
Header < boost/svg_plot/detail/svg_tag.hpp >, 122

Q

quartile_definition
Header < boost/svg_plot/svg_boxplot.hpp >, 280, 319
svg_boxplot public member functions, 300
svg_boxplot_series public member functions, 326
qurve_element
Header < boost/svg_plot/detail/svg_tag.hpp >, 148
qurve_element public member functions
class_id, 149
clip_id, 149
id, 149

R

r
Description, 179
rect_element
Header < boost/svg_plot/detail/svg_tag.hpp >, 151
rect_element public member functions
class_id, 152
clip_id, 152
id, 152
rotation
Header < boost/svg_plot/detail/svg_tag.hpp >, 166
tspan_element public member functions, 169
rounddown10
Header < boost/svg_plot/detail/auto_axes.hpp >, 87, 93
rounddown2
Header < boost/svg_plot/detail/auto_axes.hpp >, 87, 94
rounddown5
Header < boost/svg_plot/detail/auto_axes.hpp >, 87
roundup10
Header < boost/svg_plot/detail/auto_axes.hpp >, 87, 95
roundup2
Header < boost/svg_plot/detail/auto_axes.hpp >, 87, 96
roundup5
Header < boost/svg_plot/detail/auto_axes.hpp >, 87, 97

S

safe_fpt_division
Header < boost/svg_plot/detail/FP_compare.hpp >, 105
scale_axis
1-D Auto scaling Examples, 29
Header < boost/svg_plot/detail/auto_axes.hpp >, 87, 100
series_count
Header < boost/svg_plot/svg_1d_plot.hpp >, 226
svg_1d_plot_series public member functions, 228

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

series_limits_count
Header < boost/svg_plot/svg_1d_plot.hpp >, 226
svg_1d_plot_series public member functions, 228
set_ids
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_1d_plot public member functions, 212
svg_2d_plot private member functions, 271
shape
Header < boost/svg_plot/svg_1d_plot.hpp >, 226
Header < boost/svg_plot/svg_2d_plot.hpp >, 275
Header < boost/svg_plot/svg_style.hpp >, 349
plot_point_style public member functions, 350
svg_1d_plot_series public member functions, 228
svg_2d_plot_series public member functions, 278
show
Header < boost/svg_plot/detail/auto_axes.hpp >, 87, 101
show_1d_plot_settings
1-D Auto scaling Examples, 29
1-D Data Values Examples, 41
Tutorial: 2D Special Features, 59
size
close_to public member functions, 106
g_element public member functions, 126
Header < boost/svg_plot/detail/FP_compare.hpp >, 106, 107
Header < boost/svg_plot/detail/svg_tag.hpp >, 122
Header < boost/svg_plot/svg.hpp >, 190
Header < boost/svg_plot/svg_1d_plot.hpp >, 226
Header < boost/svg_plot/svg_2d_plot.hpp >, 275
Header < boost/svg_plot/svg_style.hpp >, 349
plot_point_style public member functions, 350
smallest public member functions, 107
svg public member functions, 196
svg_1d_plot_series public member functions, 228
svg_2d_plot_series public member functions, 278
size_legend_box
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot protected member functions, 225
svg_2d_plot protected member functions, 273
svg_boxplot protected member functions, 317
smallest
Header < boost/svg_plot/detail/FP_compare.hpp >, 107
smallest public member functions
size, 107
strength
close_to public member functions, 106
Header < boost/svg_plot/detail/FP_compare.hpp >, 106
strip_e0s
Header < boost/svg_plot/svg_1d_plot.hpp >, 197, 230
stroke
box_style public member functions, 345
Header < boost/svg_plot/svg_style.hpp >, 344
stroke_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 226
Header < boost/svg_plot/svg_fwd.hpp >, 337
svg_1d_plot_series public member functions, 228

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

stroke_on
Header < boost/svg_plot/svg_fwd.hpp >, 337
stroke_width
Header < boost/svg_plot/svg_fwd.hpp >, 337
svg
Header < boost/svg_plot/detail/svg_tag.hpp >, 113, 116, 119, 122, 129, 133, 144, 148, 151, 159, 164, 166
Header < boost/svg_plot/svg.hpp >, 190
svg public member functions
author, 192
boost_license_on, 192
coord_precision, 193
copyright_date, 193
copyright_holder, 193
description, 193
document_size, 194
document_title, 194
image_filename, 194
license, 194
license_on, 194, 195
size, 196
x_size, 196
y_size, 196
svg_1d_plot
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
svg_1d_plot protected member functions
clear_all, 223
clear_background, 223
clear_grids, 223
clear_legend, 223
clear_plot_background, 223
clear_points, 224
clear_title, 224
clear_x_axis, 224
clear_y_axis, 224
draw_legend, 224
draw_title, 224
draw_x_axis, 225
draw_x_axis_label, 225
place_legend_box, 225
size_legend_box, 225
svg_1d_plot public member functions
autoscale, 205
autoscale_check_limits, 205
autoscale_plusminus, 205
axes_on, 205
background_border_color, 205
background_border_width, 205
background_color, 205
boost_license_on, 205
calculate_plot_window, 206
calculate_transform, 206
coord_precision, 206
copyright_date, 206
copyright_holder, 206
data_lines_width, 206
document_title, 206
draw_axes, 206
image_border_margin, 207

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

image_border_width, 207
image_x_size, 207
image_y_size, 207
legend_background_color, 207
legend_border_color, 208
legend_box_fill_on, 208
legend_color, 208
legend_header_font_size, 208
legend_lines, 208
legend_on, 208
legend_outside, 209
legend_place, 209
legend_title, 209
legend_title_font_size, 209
legend_width, 209
license_attribution, 209
license_commercialuse, 209
license_distribution, 209
license_on, 210
license_reproduction, 210
limit_color, 210
limit_fill_color, 210
one_sd_color, 210
plot_background_color, 211
plot_border_color, 211
plot_border_width, 211
plot_window_on, 211
plot_window_x_left, 211
plot_window_x_right, 211
plot_window_y_bottom, 212
plot_window_y_top, 212
set_ids, 212
three_sd_color, 212
title, 212
title_color, 212
title_font_alignment, 212
title_font_rotation, 213
title_font_size, 213
title_on, 213
two_sd_color, 213
update_image, 214
x_autoscale, 214
x_auto_max_value, 214
x_auto_min_value, 214
x_auto_ticks, 214
x_auto_tick_interval, 214
x_axis_color, 214
x_axis_label_color, 215
x_axis_on, 215
x_axis_position, 215
x_axis_width, 215
x_df_color, 215
x_df_on, 215
x_label, 215
x_label_color, 215
x_label_font_size, 216
x_label_on, 216
x_label_units, 216

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

x_label_units_on, 216
x_label_width, 216
x_major_grid_color, 216
x_major_grid_on, 217
x_major_grid_width, 217
x_major_interval, 217
x_major_labels_side, 217
x_major_label_rotation, 217
x_major_tick, 217
x_major_tick_color, 217
x_major_tick_length, 217
x_major_tick_width, 218
x_max, 218
x_min, 218
x_minor_grid_color, 218
x_minor_grid_on, 218
x_minor_grid_width, 218
x_minor_interval, 218
x_minor_tick_color, 219
x_minor_tick_length, 219
x_minor_tick_width, 219
x_min_ticks, 218
x_num_minor_ticks, 219
x_plusminus_color, 219
x_plusminus_on, 219
x_prefix, 219
x_separator, 219
x_steps, 220
x_suffix, 220
x_ticks_down_on, 220
x_ticks_on_window_or_axis, 220
x_ticks_up_on, 220
x_ticks_values_color, 220
x_ticks_values_font_size, 220
x_ticks_values_ioflags, 220
x_ticks_values_precision, 221
x_tight, 221
x_values_color, 221
x_values_font_size, 221
x_values_ioflags, 222
x_values_on, 222
x_values_precision, 222
x_values_rotation, 222
x_value_font_size, 221
x_value_ioflags, 221
x_value_precision, 221
x_with_zero, 222
y_axis_color, 222
y_axis_on, 222
y_label, 222
y_labels_strip_e0s, 223
y_label_color, 223
y_label_units, 223
y_minor_interval, 223
svg_1d_plot_series
Header < boost/svg_plot/svg_1d_plot.hpp >, 226
svg_1d_plot_series public member functions
bezier_on, 227

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

fill_color, 227
line_on, 227
line_width, 227
series_count, 228
series_limits_count, 228
shape, 228
size, 228
stroke_color, 228
symbols, 229
svg_2d_plot
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot private member functions
calculate_plot_window, 270
draw_bars, 270
draw_histogram, 270
draw_plot_lines, 271
draw_plot_points, 271
draw_y_axis, 271
draw_y_axis_label, 271
set_ids, 271
update_image, 271
svg_2d_plot protected member functions
clear_all, 271
clear_background, 272
clear_grids, 272
clear_legend, 272
clear_plot_background, 272
clear_points, 272
clear_title, 272
clear_x_axis, 272
clear_y_axis, 272
draw_legend, 272
draw_title, 273
draw_x_axis, 273
draw_x_axis_label, 273
place_legend_box, 273
size_legend_box, 273
svg_2d_plot public member functions
autoscale, 240
autoscale_check_limits, 240
autoscale_plusminus, 241
axes_on, 241
background_border_color, 241
background_border_width, 241
background_color, 241
boost_license_on, 241
coord_precision, 241
copyright_date, 241
copyright_holder, 241
data_lines_width, 242
document_title, 242
image_border_margin, 242
image_border_width, 242
image_x_size, 243
image_y_size, 243
legend_background_color, 243
legend_border_color, 243
legend_box_fill_on, 243

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

legend_color, 243
legend_header_font_size, 244
legend_lines, 244
legend_on, 244
legend_outside, 244
legend_place, 244
legend_title, 244
legend_title_font_size, 244
legend_width, 245
license_attribution, 245
license_commercialuse, 245
license_distribution, 245
license_on, 245
license_reproduction, 245
limit_color, 245
limit_fill_color, 245
one_sd_color, 245
plot_background_color, 246
plot_border_color, 246
plot_border_width, 247
plot_window_on, 247
plot_window_x_left, 247
plot_window_x_right, 247
plot_window_y_bottom, 247
plot_window_y_top, 247
three_sd_color, 247
title, 247
title_color, 248
title_font_alignment, 248
title_font_rotation, 248
title_font_size, 248
title_on, 249
two_sd_color, 249
xy_autoscale, 258
xy_values_on, 258
x_autoscale, 250
x_auto_max_value, 249
x_auto_min_value, 249
x_auto_ticks, 249
x_auto_tick_interval, 249
x_axis_color, 250
x_axis_label_color, 250
x_axis_on, 250
x_axis_position, 250
x_axis_width, 250
x_df_color, 251
x_df_on, 251
x_label, 251
x_label_color, 251
x_label_font_size, 251
x_label_on, 251
x_label_units, 251
x_label_units_on, 252
x_label_width, 252
x_major_grid_color, 252
x_major_grid_on, 252
x_major_grid_width, 252
x_major_interval, 252

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

x_major_labels_side, 253
x_major_label_rotation, 252
x_major_tick, 253
x_major_tick_color, 253
x_major_tick_length, 253
x_major_tick_width, 253
x_max, 253
x_min, 253
x_minor_grid_color, 254
x_minor_grid_on, 254
x_minor_grid_width, 254
x_minor_interval, 254
x_minor_tick_color, 254
x_minor_tick_length, 254
x_minor_tick_width, 254
x_min_ticks, 253
x_num_minor_ticks, 254
x_plusminus_color, 255
x_plusminus_on, 255
x_prefix, 255
x_separator, 255
x_steps, 255
x_suffix, 255
x_ticks_down_on, 255
x_ticks_on_window_or_axis, 255
x_ticks_up_on, 256
x_ticks_values_color, 256
x_ticks_values_font_size, 256
x_ticks_values_ioflags, 256
x_ticks_values_precision, 256
x_tight, 256
x_values_color, 257
x_values_font_size, 257
x_values_ioflags, 257
x_values_on, 257
x_values_precision, 257
x_values_rotation, 258
x_value_font_size, 256
x_value_ioflags, 257
x_value_precision, 257
x_with_zero, 258
y_autoscale, 259
y_axis_color, 259
y_axis_label_color, 259
y_axis_on, 259
y_axis_position, 259
y_axis_value_color, 259
y_axis_width, 260
y_df_color, 260
y_df_on, 260
y_label, 260
y_labels_strip_e0s, 262
y_label_axis, 260
y_label_color, 261
y_label_font_size, 261
y_label_on, 261
y_label_units, 261
y_label_units_on, 261

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

y_label_width, 262
y_major_grid_color, 262
y_major_grid_on, 262
y_major_grid_width, 263
y_major_interval, 263
y_major_labels_side, 263
y_major_label_rotation, 263
y_major_tick_color, 264
y_major_tick_length, 264
y_major_tick_width, 264
y_max, 264
y_min, 264
y_minor_grid_color, 264
y_minor_grid_on, 264
y_minor_grid_width, 265
y_minor_interval, 265
y_minor_tick_color, 265
y_minor_tick_length, 265
y_minor_tick_width, 265
y_num_minor_ticks, 265
y_plusminus_color, 266
y_plusminus_on, 266
y_prefix, 266
y_separator, 266
y_suffix, 266
y_ticks_left_on, 267
y_ticks_on_window_or_axis, 267
y_ticks_right_on, 267
y_ticks_values_color, 267
y_ticks_values_font_size, 268
y_ticks_values_ioflags, 268
y_ticks_values_precision, 268
y_values_color, 269
y_values_font_size, 269
y_values_ioflags, 269
y_values_on, 269
y_values_precision, 270
y_values_rotation, 270
y_value_ioflags, 268
y_value_precision, 268
svg_2d_plot_series
Header < boost/svg_plot/svg_2d_plot.hpp >, 275
svg_2d_plot_series public member functions
bar_opt, 276
bar_width, 276
bezier_on, 277
limits_count, 277
line_on, 277
line_style, 277
line_width, 278
shape, 278
size, 278
values_count, 278
svg_boxplot
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_boxplot protected member functions
clear_background, 316
clear_grids, 316

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

clear_legend, 316
clear_plot_background, 316
clear_points, 316
clear_title, 316
clear_x_axis, 316
clear_y_axis, 316
draw_legend, 316
place_legend_box, 317
size_legend_box, 317
svg_boxplot public member functions
autoscale, 288
autoscale_check_limits, 288
autoscale_plusminus, 288
axes_on, 288
axis_color, 288
axis_width, 289
background_border_color, 289
background_border_width, 289
background_color, 289
boost_license_on, 289
box_border, 289
box_fill, 290
box_width, 290
calculate_plot_window, 290
clear_all, 290
coord_precision, 290
copyright_date, 290
copyright_holder, 290
data_lines_width, 291
document_title, 291
draw_title, 292
draw_x_axis, 292
draw_x_axis_label, 292
draw_y_axis, 292
draw_y_axis_label, 292
extreme_outlier_color, 293
extreme_outlier_fill, 293
extreme_outlier_shape, 293
extreme_outlier_size, 293
extreme_outlier_values_on, 293
image_border_margin, 294
image_border_width, 294
image_x_size, 294
image_y_size, 294
legend_background_color, 294
legend_border_color, 294
legend_box_fill_on, 294
legend_color, 295
legend_header_font_size, 295
legend_lines, 295
legend_on, 295
legend_outside, 295
legend_place, 295
legend_title, 295
legend_title_font_size, 296
legend_width, 296
license_attribution, 296
license_commercialuse, 296

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

license_distribution, 296
license_on, 296
license_reproduction, 296
limit_color, 296
limit_fill_color, 297
median_color, 297
median_values_on, 297
median_width, 297
one_sd_color, 297
outlier_color, 297
outlier_fill, 298
outlier_shape, 298
outlier_size, 298
outlier_values_on, 298
plot_background_color, 299
plot_border_color, 299
plot_border_width, 299
plot_window_on, 299
plot_window_x_left, 300
plot_window_x_right, 300
plot_window_y_bottom, 300
plot_window_y_top, 300
quartile_definition, 300
three_sd_color, 300
title, 300
title_color, 301
title_font_alignment, 301
title_font_rotation, 301
title_font_size, 301
title_on, 302
two_sd_color, 302
update_image, 302
whisker_length, 303
x_autoscale, 303
x_auto_max_value, 303
x_auto_min_value, 303
x_auto_ticks, 303
x_auto_tick_interval, 303
x_axis_color, 304
x_axis_label_color, 304
x_axis_on, 304
x_axis_position, 304
x_axis_width, 304
x_df_color, 304
x_df_on, 304
x_label, 305
x_label_color, 305
x_label_font_size, 305
x_label_on, 305
x_label_text, 306
x_label_units, 306
x_label_units_on, 306
x_label_width, 306
x_major_grid_color, 306
x_major_grid_on, 306
x_major_grid_width, 306
x_major_interval, 306
x_major_labels, 307

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

x_major_labels_side, 307
x_major_label_rotation, 307
x_major_tick, 307
x_major_tick_color, 307
x_major_tick_length, 307
x_major_tick_width, 307
x_max, 307
x_min, 308
x_minor_grid_color, 308
x_minor_grid_on, 308
x_minor_grid_width, 308
x_minor_interval, 308
x_minor_tick_color, 308
x_minor_tick_length, 308
x_minor_tick_width, 309
x_min_ticks, 308
x_num_minor_ticks, 309
x_plusminus_color, 309
x_plusminus_on, 309
x_prefix, 309
x_separator, 309
x_steps, 309
x_suffix, 309
x_ticks_down_on, 310
x_ticks_on_window_or_axis, 310
x_ticks_up_on, 310
x_ticks_values_color, 310
x_ticks_values_font_size, 311
x_ticks_values_ioflags, 311
x_ticks_values_precision, 311
x_tick_color, 310
x_tick_length, 310
x_tight, 311
x_values_color, 312
x_values_font_size, 312
x_values_ioflags, 312
x_values_on, 312
x_values_precision, 312
x_values_rotation, 312
x_value_font_size, 311
x_value_ioflags, 311
x_value_precision, 311
x_with_zero, 312
y_autoscale, 313
y_axis_color, 313
y_axis_on, 313
y_axis_position, 313
y_label, 314
y_labels_strip_e0s, 314
y_label_color, 314
y_label_text, 314
y_label_units, 314
y_minor_interval, 315
svg_boxplot_series
Header < boost/svg_plot/svg_boxplot.hpp >, 319
svg_boxplot_series public member functions
axis_color, 320
axis_width, 321

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

box_border, 321
box_fill, 321
box_width, 322
calculate_quantiles, 322
extreme_outlier_color, 322
extreme_outlier_fill, 323
extreme_outlier_shape, 323
extreme_outlier_size, 323
max_whisker_color, 323
max_whisker_width, 323
median_color, 324
median_width, 324
min_whisker_color, 324
min_whisker_width, 325
outlier_color, 325
outlier_fill, 325
outlier_shape, 325
outlier_size, 326
quartile_definition, 326
title, 326
whisker_length, 327
svg_color
Header < boost/svg_plot/svg_color.hpp >, 328
svg_color public construct/copy/destruct
my_blank, 328
svg_element
Header < boost/svg_plot/detail/svg_tag.hpp >, 113, 116, 119, 122, 129, 133, 144, 148, 151, 155, 159, 166
svg_element public member functions
class_id, 156
clip_id, 156
id, 156
svg_style
Header < boost/svg_plot/svg_style.hpp >, 352
symbols
Header < boost/svg_plot/svg_1d_plot.hpp >, 226
svg_1d_plot_series public member functions, 229

T

t
Description, 178
text
Header < boost/svg_plot/detail/svg_tag.hpp >, 159, 166
text_element public member functions, 162
tspan_element public member functions, 170
text_element
Header < boost/svg_plot/detail/svg_tag.hpp >, 159
text_element public member functions
alignment, 160
class_id, 160
clip_id, 161
id, 161
text, 162
text_element_text
Header < boost/svg_plot/detail/svg_tag.hpp >, 164
text_length
Header < boost/svg_plot/detail/svg_tag.hpp >, 166
tspan_element public member functions, 170

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

text_parent
Header < boost/svg_plot/detail/svg_tag.hpp >, 164, 165, 166
text_style
Header < boost/svg_plot/svg_style.hpp >, 355
three_sd_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 212
svg_2d_plot public member functions, 247
svg_boxplot public member functions, 300
ticks_labels_style
Header < boost/svg_plot/svg_style.hpp >, 358
ticks_labels_style public member functions
label_length, 359
longest_label, 359
title
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280, 319
svg_1d_plot public member functions, 212
svg_2d_plot public member functions, 247
svg_boxplot public member functions, 300
svg_boxplot_series public member functions, 326
title_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 212
svg_2d_plot public member functions, 248
svg_boxplot public member functions, 301
title_font_alignment
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 212
svg_2d_plot public member functions, 248
svg_boxplot public member functions, 301
title_font_rotation
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 213
svg_2d_plot public member functions, 248
svg_boxplot public member functions, 301
title_font_size
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 213
svg_2d_plot public member functions, 248
svg_boxplot public member functions, 301
title_on
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 213
svg_2d_plot public member functions, 249

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

svg_boxplot public member functions, 302
ts
Description, 367
tspan_element
Header < boost/svg_plot/detail/svg_tag.hpp >, 166
tspan_element public member functions
class_id, 167
clip_id, 167
dx, 168
dy, 168
fill_color, 168
fill_on, 168
font_size, 168
id, 169
rotation, 169
text, 170
text_length, 170
use_style, 170
x, 171
y, 171
Tutorial: 2D Special Features
show_1d_plot_settings, 59
two_sd_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 213
svg_2d_plot public member functions, 249
svg_boxplot public member functions, 302
types
Header < boost/svg_plot/uncertain.hpp >, 371
unc public member functions, 372
t_or_b
Header < boost/svg_plot/show_1d_settings.hpp >, 182

U

unc
Header < boost/svg_plot/uncertain.hpp >, 371
unc public member functions
deg_free, 372
types, 372
uncertainty, 372
value, 372
uncertainty
Header < boost/svg_plot/uncertain.hpp >, 371
unc public member functions, 372
unc_of
Header < boost/svg_plot/uncertain.hpp >, 370, 376, 377
update_image
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 214
svg_2d_plot private member functions, 271
svg_boxplot public member functions, 302
use_style
Header < boost/svg_plot/detail/svg_tag.hpp >, 166

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

tspan_element public member functions, 170

V

value
Header < boost/svg_plot/uncertain.hpp >, 371
unc public member functions, 372
values_count
Header < boost/svg_plot/svg_2d_plot.hpp >, 275
svg_2d_plot_series public member functions, 278
value_of
Header < boost/svg_plot/uncertain.hpp >, 370, 380, 381
value_style
Header < boost/svg_plot/svg_style.hpp >, 361

W

whisker_length
Header < boost/svg_plot/svg_boxplot.hpp >, 280, 319
svg_boxplot public member functions, 303
svg_boxplot_series public member functions, 327
width
axis_line_style public member functions, 341
bar_style public member functions, 343
box_style public member functions, 345
Demonstration of using 1D data that includes information about its Uncertainty, 52
Header < boost/svg_plot/svg_style.hpp >, 340, 342, 344, 347
plot_line_style public member functions, 348
width_on
Header < boost/svg_plot/svg_fwd.hpp >, 337

X

x
Header < boost/svg_plot/detail/svg_tag.hpp >, 166
tspan_element public member functions, 171
xy_autoscale
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 258
xy_values_on
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 258
x_autoscale
1-D Auto scaling Examples, 29
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 214
svg_2d_plot public member functions, 250
svg_boxplot public member functions, 303
x_auto_max_value
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 214
svg_2d_plot public member functions, 249
svg_boxplot public member functions, 303
x_auto_min_value
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 214
svg_2d_plot public member functions, 249
svg_boxplot public member functions, 303
x_auto_ticks
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 214
svg_2d_plot public member functions, 249
svg_boxplot public member functions, 303
x_auto_tick_interval
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 214
svg_2d_plot public member functions, 249
svg_boxplot public member functions, 303
x_axis_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 214
svg_2d_plot public member functions, 250
svg_boxplot public member functions, 304
x_axis_label_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 215
svg_2d_plot public member functions, 250
svg_boxplot public member functions, 304
x_axis_on
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 215
svg_2d_plot public member functions, 250
svg_boxplot public member functions, 304
x_axis_position
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 215
svg_2d_plot public member functions, 250
svg_boxplot public member functions, 304
x_axis_width
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 215
svg_2d_plot public member functions, 250
svg_boxplot public member functions, 304
x_df_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 215

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

svg_2d_plot public member functions, 251
svg_boxplot public member functions, 304
x_df_on
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 215
svg_2d_plot public member functions, 251
svg_boxplot public member functions, 304
x_label
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 215
svg_2d_plot public member functions, 251
svg_boxplot public member functions, 305
x_label_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 215
svg_2d_plot public member functions, 251
svg_boxplot public member functions, 305
x_label_font_size
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 216
svg_2d_plot public member functions, 251
svg_boxplot public member functions, 305
x_label_on
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 216
svg_2d_plot public member functions, 251
svg_boxplot public member functions, 305
x_label_text
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_boxplot public member functions, 306
x_label_units
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 216
svg_2d_plot public member functions, 251
svg_boxplot public member functions, 306
x_label_units_on
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 216
svg_2d_plot public member functions, 252
svg_boxplot public member functions, 306
x_label_width
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

svg_1d_plot public member functions, 216
svg_2d_plot public member functions, 252
svg_boxplot public member functions, 306
x_major_grid_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 216
svg_2d_plot public member functions, 252
svg_boxplot public member functions, 306
x_major_grid_on
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 217
svg_2d_plot public member functions, 252
svg_boxplot public member functions, 306
x_major_grid_width
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 217
svg_2d_plot public member functions, 252
svg_boxplot public member functions, 306
x_major_interval
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 217
svg_2d_plot public member functions, 252
svg_boxplot public member functions, 306
x_major_labels
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_boxplot public member functions, 307
x_major_labels_side
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 217
svg_2d_plot public member functions, 253
svg_boxplot public member functions, 307
x_major_label_rotation
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 217
svg_2d_plot public member functions, 252
svg_boxplot public member functions, 307
x_major_tick
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 217
svg_2d_plot public member functions, 253
svg_boxplot public member functions, 307
x_major_tick_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 217
svg_2d_plot public member functions, 253
svg_boxplot public member functions, 307
x_major_tick_length
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 217
svg_2d_plot public member functions, 253
svg_boxplot public member functions, 307
x_major_tick_width
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 218
svg_2d_plot public member functions, 253
svg_boxplot public member functions, 307
x_max
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 218
svg_2d_plot public member functions, 253
svg_boxplot public member functions, 307
x_min
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 218
svg_2d_plot public member functions, 253
svg_boxplot public member functions, 308
x_minor_grid_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 218
svg_2d_plot public member functions, 254
svg_boxplot public member functions, 308
x_minor_grid_on
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 218
svg_2d_plot public member functions, 254
svg_boxplot public member functions, 308
x_minor_grid_width
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 218
svg_2d_plot public member functions, 254
svg_boxplot public member functions, 308
x_minor_interval
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 218

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

svg_2d_plot public member functions, 254
svg_boxplot public member functions, 308
x_minor_tick_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 219
svg_2d_plot public member functions, 254
svg_boxplot public member functions, 308
x_minor_tick_length
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 219
svg_2d_plot public member functions, 254
svg_boxplot public member functions, 308
x_minor_tick_width
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 219
svg_2d_plot public member functions, 254
svg_boxplot public member functions, 309
x_min_ticks
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 218
svg_2d_plot public member functions, 253
svg_boxplot public member functions, 308
x_num_minor_ticks
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 219
svg_2d_plot public member functions, 254
svg_boxplot public member functions, 309
x_plusminus_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 219
svg_2d_plot public member functions, 255
svg_boxplot public member functions, 309
x_plusminus_on
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 219
svg_2d_plot public member functions, 255
svg_boxplot public member functions, 309
x_prefix
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 219
svg_2d_plot public member functions, 255
svg_boxplot public member functions, 309

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

x_separator
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 219
svg_2d_plot public member functions, 255
svg_boxplot public member functions, 309
x_size
Header < boost/svg_plot/svg.hpp >, 190
Header < boost/svg_plot/svg_fwd.hpp >, 337
svg public member functions, 196
x_steps
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 220
svg_2d_plot public member functions, 255
svg_boxplot public member functions, 309
x_suffix
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 220
svg_2d_plot public member functions, 255
svg_boxplot public member functions, 309
x_ticks_down_on
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 220
svg_2d_plot public member functions, 255
svg_boxplot public member functions, 310
x_ticks_on_window_or_axis
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 220
svg_2d_plot public member functions, 255
svg_boxplot public member functions, 310
x_ticks_up_on
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 220
svg_2d_plot public member functions, 256
svg_boxplot public member functions, 310
x_ticks_values_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 220
svg_2d_plot public member functions, 256
svg_boxplot public member functions, 310
x_ticks_values_font_size
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 220

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

svg_2d_plot public member functions, 256
svg_boxplot public member functions, 311
x_ticks_values_ioflags
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 220
svg_2d_plot public member functions, 256
svg_boxplot public member functions, 311
x_ticks_values_precision
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 221
svg_2d_plot public member functions, 256
svg_boxplot public member functions, 311
x_tick_color
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_boxplot public member functions, 310
x_tick_length
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_boxplot public member functions, 310
x_tight
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 221
svg_2d_plot public member functions, 256
svg_boxplot public member functions, 311
x_values_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 221
svg_2d_plot public member functions, 257
svg_boxplot public member functions, 312
x_values_font_size
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 221
svg_2d_plot public member functions, 257
svg_boxplot public member functions, 312
x_values_ioflags
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 222
svg_2d_plot public member functions, 257
svg_boxplot public member functions, 312
x_values_on
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 222
svg_2d_plot public member functions, 257
svg_boxplot public member functions, 312
x_values_precision

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 222
svg_2d_plot public member functions, 257
svg_boxplot public member functions, 312
x_values_rotation
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 222
svg_2d_plot public member functions, 258
svg_boxplot public member functions, 312
x_value_font_size
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 221
svg_2d_plot public member functions, 256
svg_boxplot public member functions, 311
x_value_ioflags
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 221
svg_2d_plot public member functions, 257
svg_boxplot public member functions, 311
x_value_precision
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 221
svg_2d_plot public member functions, 257
svg_boxplot public member functions, 311
x_with_zero
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 222
svg_2d_plot public member functions, 258
svg_boxplot public member functions, 312

Y

y
Header < boost/svg_plot/detail/svg_tag.hpp >, 166
tspan_element public member functions, 171
y_autoscale
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_2d_plot public member functions, 259
svg_boxplot public member functions, 313
y_axis_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 222
svg_2d_plot public member functions, 259

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

svg_boxplot public member functions, 313
y_axis_label_color
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 259
y_axis_on
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 222
svg_2d_plot public member functions, 259
svg_boxplot public member functions, 313
y_axis_position
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_2d_plot public member functions, 259
svg_boxplot public member functions, 313
y_axis_value_color
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 259
y_axis_width
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 260
y_df_color
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 260
y_df_on
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 260
y_label
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 222
svg_2d_plot public member functions, 260
svg_boxplot public member functions, 314
y_labels_strip_e0s
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 223
svg_2d_plot public member functions, 262
svg_boxplot public member functions, 314
y_label_axis
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 260
y_label_color
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 223
svg_2d_plot public member functions, 261
svg_boxplot public member functions, 314
y_label_font_size
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 261
y_label_on
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 261

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

y_label_text
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_boxplot public member functions, 314
y_label_units
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 223
svg_2d_plot public member functions, 261
svg_boxplot public member functions, 314
y_label_units_on
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 261
y_label_width
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 262
y_major_grid_color
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 262
y_major_grid_on
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 262
y_major_grid_width
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 263
y_major_interval
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 263
y_major_labels_side
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 263
y_major_label_rotation
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 263
y_major_tick_color
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 264
y_major_tick_length
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 264
y_major_tick_width
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 264
y_max
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 264
y_min
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 264
y_minor_grid_color
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 264
y_minor_grid_on
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 264
y_minor_grid_width
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 265

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

y_minor_interval
Header < boost/svg_plot/svg_1d_plot.hpp >, 198
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
Header < boost/svg_plot/svg_boxplot.hpp >, 280
svg_1d_plot public member functions, 223
svg_2d_plot public member functions, 265
svg_boxplot public member functions, 315
y_minor_tick_color
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 265
y_minor_tick_length
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 265
y_minor_tick_width
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 265
y_num_minor_ticks
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 265
y_plusminus_color
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 266
y_plusminus_on
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 266
y_prefix
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 266
y_separator
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 266
y_size
Header < boost/svg_plot/svg.hpp >, 190
Header < boost/svg_plot/svg_fwd.hpp >, 337
svg public member functions, 196
y_suffix
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 266
y_ticks_left_on
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 267
y_ticks_on_window_or_axis
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 267
y_ticks_right_on
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 267
y_ticks_values_color
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 267
y_ticks_values_font_size
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 268
y_ticks_values_ioflags
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 268
y_ticks_values_precision
Header < boost/svg_plot/svg_2d_plot.hpp >, 231

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

svg_2d_plot public member functions, 268
y_values_color
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 269
y_values_font_size
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 269
y_values_ioflags
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 269
y_values_on
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 269
y_values_precision
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 270
y_values_rotation
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 270
y_value_ioflags
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 268
y_value_precision
Header < boost/svg_plot/svg_2d_plot.hpp >, 231
svg_2d_plot public member functions, 268

Important

This is not (yet) an official Boost library. It was a Google Summer of Code project (2007) whose mentor organization was Boost. It remains a library under construction, the code is quite functional, but interfaces, library structure, and names may still be changed without notice.

Note

Comments and suggestions (even bugs) to Paul.A.Bristow (at) hetp (dot) u-net (dot) com or Jake Voytko at jakevoytko (at) gmail (dot) com

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/