Incanter Chart Customizations

Gregory Raven

2013-02-10 Sunday

Contents

1 Introduction

The Incanter “charts” namespace contains many functions for plotting of data. The functions use the Clojure Java interop to create instances and call methods from the JFreeChart Java library. JFreeChart is free software and includes excellent support for a wide variety of charting requirements.

Although Incanter has good support for most chart types commonly used for data analysis, it does not support many features available in JFreeChart. The JFreeChart has hundreds of objects and methods in its API. Supporting a large chunk of them within the Incanter charts namespace would be a huge task.

This is where one of the fundamental design goals of Clojure comes to the rescue. Via the “Clojure Java Interop” the entire JFreeChart API becomes available to the user. A process that can be used is to start with a basic chart generated from an Incanter charts function, and then add enhancements to this chart using the Java Interop.

This has proven to be mostly practical. A barrier to this process is the large size of the JFreeChart API. The API documentation is quite good and is available at the JFreeChart website. A “Developer Guide” is available for $65 USD and is recommended if you spend any amount of time working directly with the API. A set of example code, all pure Java, is also available.

The purpose of this document is to demonstrate a few simple method calls to the JFreeChart API to make various changes and enhancements to Incanter charts. This should save many hours of time digging through the API and example code to find how to make these changes.

What follows is a series of examples starting with the xy-plot that builds new features one step at a time. The code examples were built up within org-mode ‘src blocks’. Namespace and use declarations are shown only once and carry through to the next block. JFreeChart imports are shown as required. The code is evaluated sequentially within the blocks, in a manner similar to sequentially entering commands using the REPL.

Although the examples start with the xy-plot, the general principles should be applicable to most of the other charts.

1.1 Versions Used

The code used in this document was evaluated using version 1.4.0 Clojure and 1.4.1 Incanter. A big Thank You to the Clojure and Incanter Development Teams! It is a great language and a fantastic computational system.

2 Preliminaries

This is code to declare the Clojure namespace and define vars of simple data vectors used in the examples. Incanter is included via the :use function.

The “data” is basic Clojure vectors. “xs” is a vector of x-axis points. “ys1” is a vector of y-axis points, and “ys2” is another set of y-axis points.

```
(ns jfreechart-tests.core
    (:use [incanter core charts pdf]))
(def xs [1.0 11.0 25.0 33.0 49.0 100.])
(def ys1 [2.2 3.8 4.1 2.2 2.5 3.5])
(def ys2 [1.0 3.5 4.7 3.1 1.4 4.5])
```

3 The Incanter xy-plot Function

Evaluating the following code generates a standard Incanter xy-plot with most of the options used. The add-lines function is used to add a second set of data to the chart.

Note that the save-pdf function is used for purposes of inserting the chart into this document. For immediate viewing of the chart use the view function from the Incanter core namespace.

```
(def xy1 (xy-plot xs ys1 :x-label "The x axis" :y-label "The y axis" :title "XY Plot"
                         :legend true :series-label "The 1st Series" :points true))
(add-lines xy1 xs ys2 :series-label "The 2nd Series" :points true)
(save-pdf xy1 "./src/jfreechart_tests/xy-standard.pdf")
```

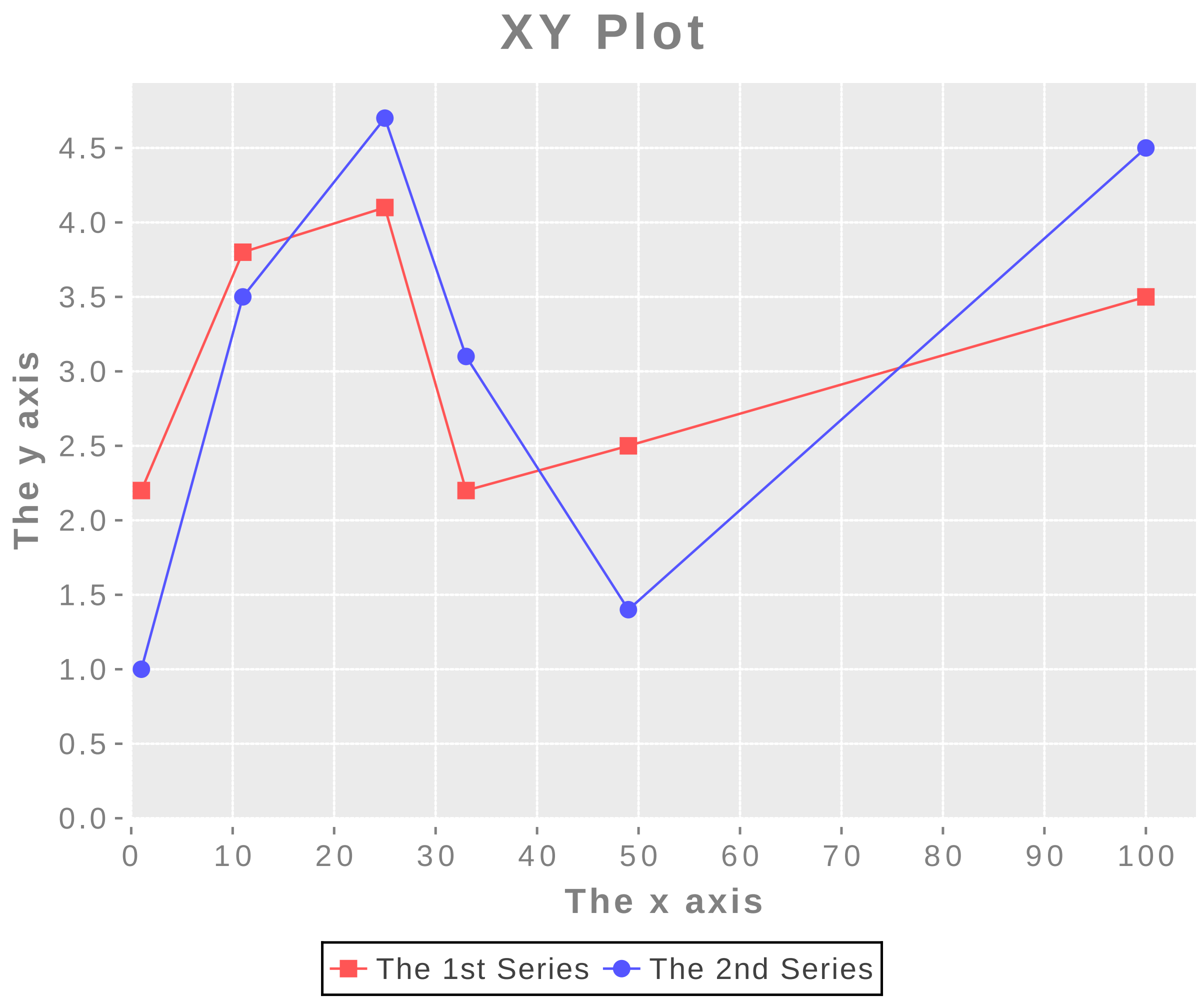

Figure 1: **An example of the chart generated by the xy-plot function.**

The documentation for the Incanter charts xy-plot function:

http://liebke.github.com/incanter/charts-api.html#incanter.charts/xy-plot

4 Make the Axes Visible

The chart does not have visible axes. The following code shows how the Clojure Java-interop facility is used to invoke the method which sets the axes to visible. The method is invoked once each for the domain (x) and range (y) axes.

Note how the plot object is extracted from the chart using the .getPlot method. The var “plot” is used hereafter to simplify the code.

```
(def plot (.getPlot xy1))
(.setAxisLineVisible (.getDomainAxis plot) true)
(.setAxisLineVisible (.getRangeAxis plot) true)
(save-pdf xy1 "./src/jfreechart_tests/xy-axes.pdf")
```

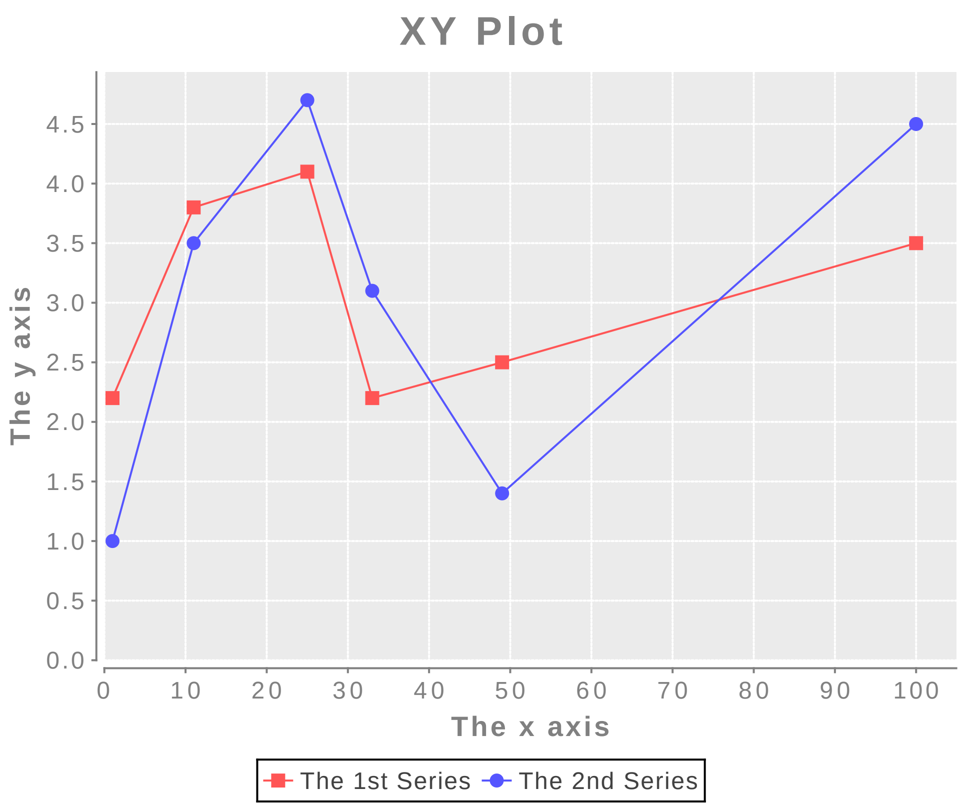

Figure 2: **The chart with the default axes visible.**

Here is the applicable documentation for the methods used in the above code:

getPlot

http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/JFreeChart.html

setAxisLineVisible

http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/Axis.html

getDomainAxis and getRangeAxis

http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/XYPlot.html

5 Set the Axis “Tick Unit”

It is possible to adjust the “Tick Unit”, which is the spacing of the tick marks on the axis. Here is an example which shows how this can be achieved. Both the x and y axix ticks are changed.

Note how the Java Interop is used to create a NumberTickUnit object. The “.” is placed after the object name to invoke the object’s constructor. The constructor parameters follow.

```
(.setTickUnit (.getDomainAxis plot) (org.jfree.chart.axis.NumberTickUnit. 5.0))
(.setTickUnit (.getRangeAxis plot) (org.jfree.chart.axis.NumberTickUnit. 1.0))
(save-pdf xy1 "./src/jfreechart_tests/xy-axes-ticks.pdf")
```

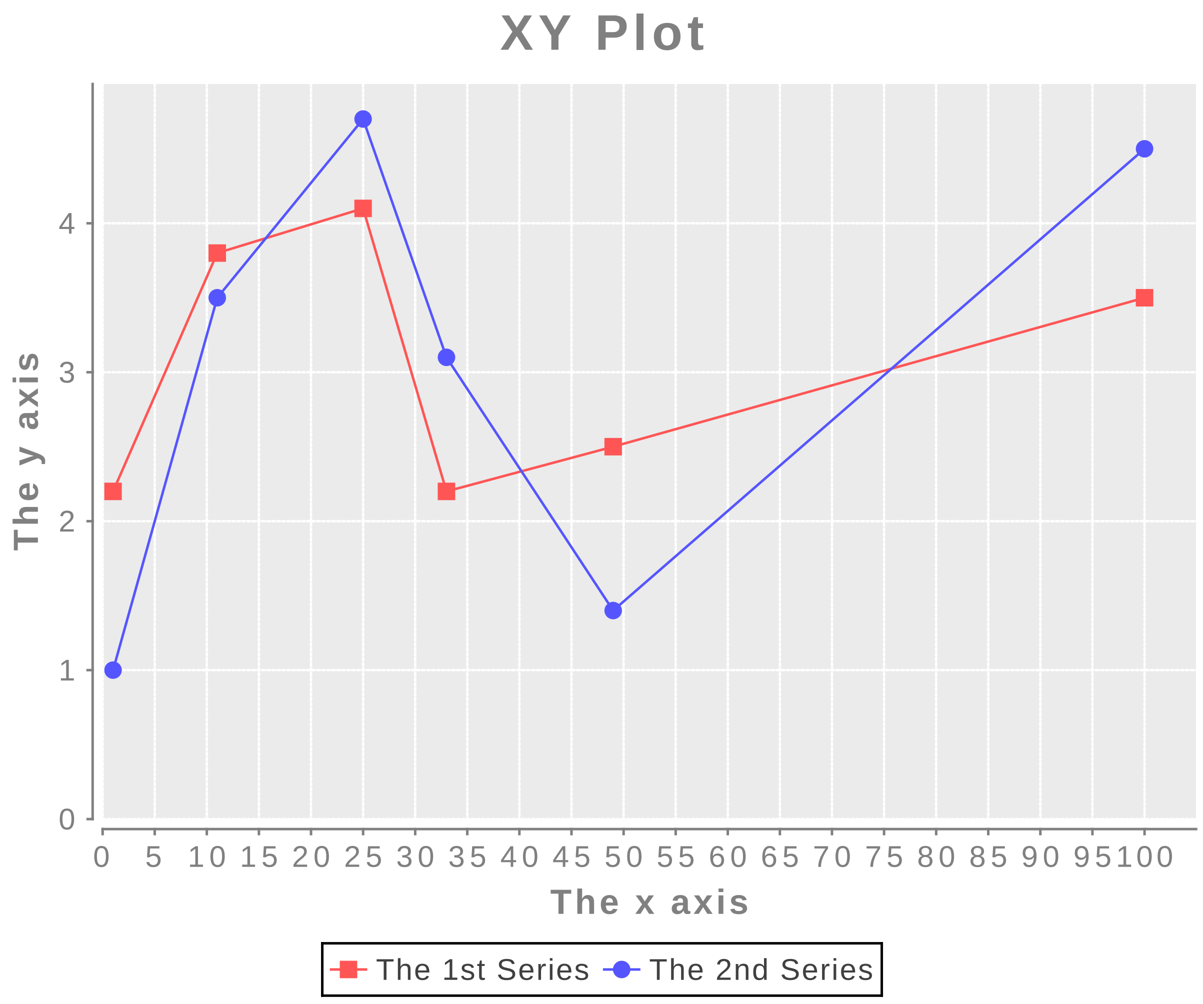

Figure 3: **The chart with the default axes active.**

Here is the applicable documentation for the Class and method introduced in the above code:

setTickUnit

http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/NumberAxis.html

NumberTickUnit

http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/NumberTickUnit.html

6 Change to a Logarithmic Axis Style 1

The xy-plot default is a linear axis. The axis (or axes) can be changed to logarithmic. In fact, there are two styles of logarithmic axis styles available.

Note the import of the LogarithmicAxis class:

http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/LogarithmicAxis.html

```
(import (org.jfree.chart.axis LogarithmicAxis))
(.setDomainAxis plot (LogarithmicAxis. "Logarithmic Axis Style 1"))
(save-pdf xy1 "./src/jfreechart_tests/xy-axes-log1.pdf")
```

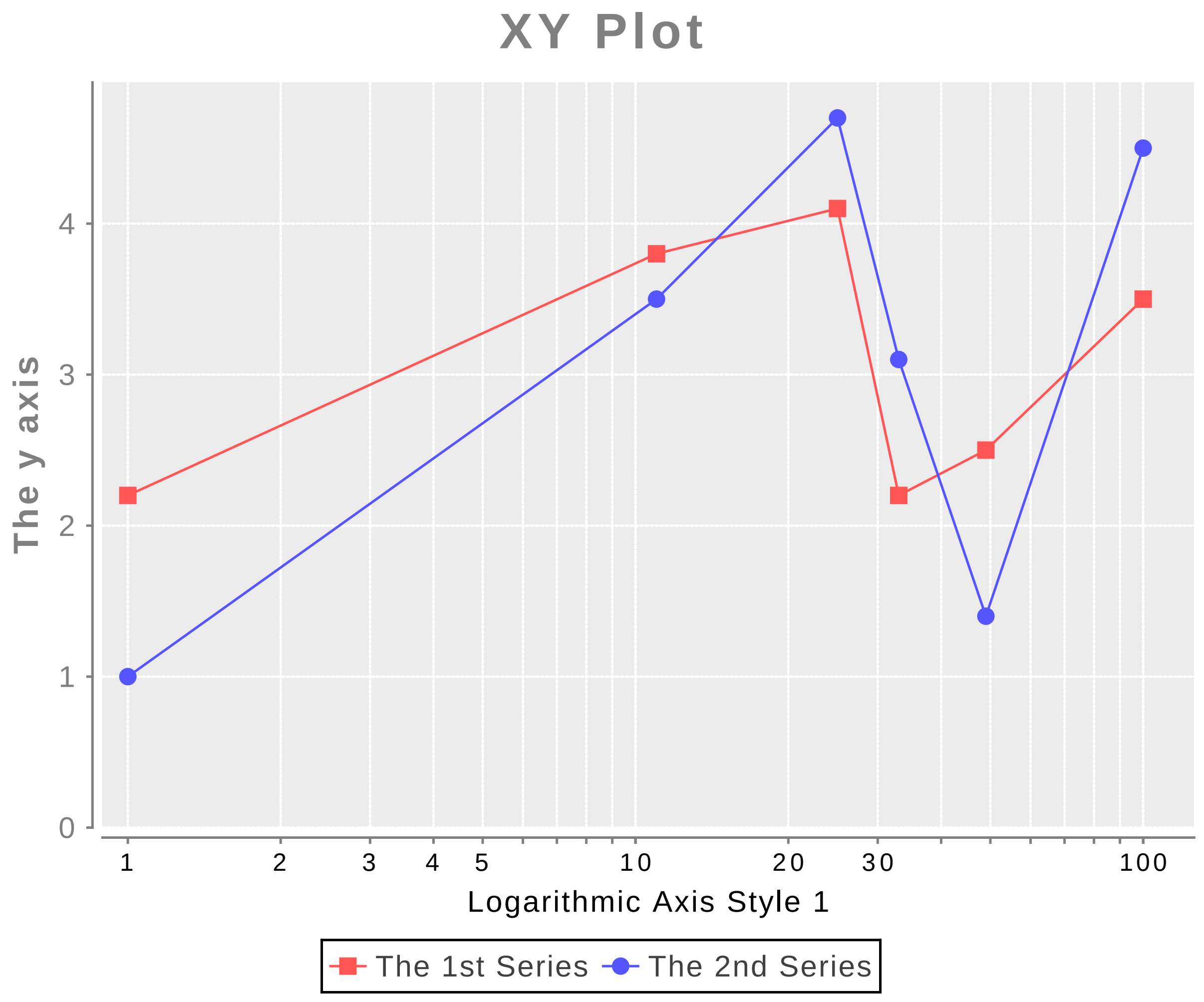

Figure 4: **The chart with the x-axis changed to logarithmic style 1.**

Here is the applicable documentation for the Class introduced in the above code:

LogarithmicAxis

http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/LogarithmicAxis.html

7 Change to a Logarithmic Axis Style 2

The xy-plot default is a linear axis. The axis (or axes) can be changed to logarithmic. In fact, there are two styles of logarithmic axis styles available.

Note the import of the LogAxis class:

http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/LogAxis.html

```
(import (org.jfree.chart.axis LogAxis))
(.setDomainAxis plot (LogAxis. "Logarithmic Axis Style 2"))
(save-pdf xy1 "./src/jfreechart_tests/xy-axes-log2.pdf")
```

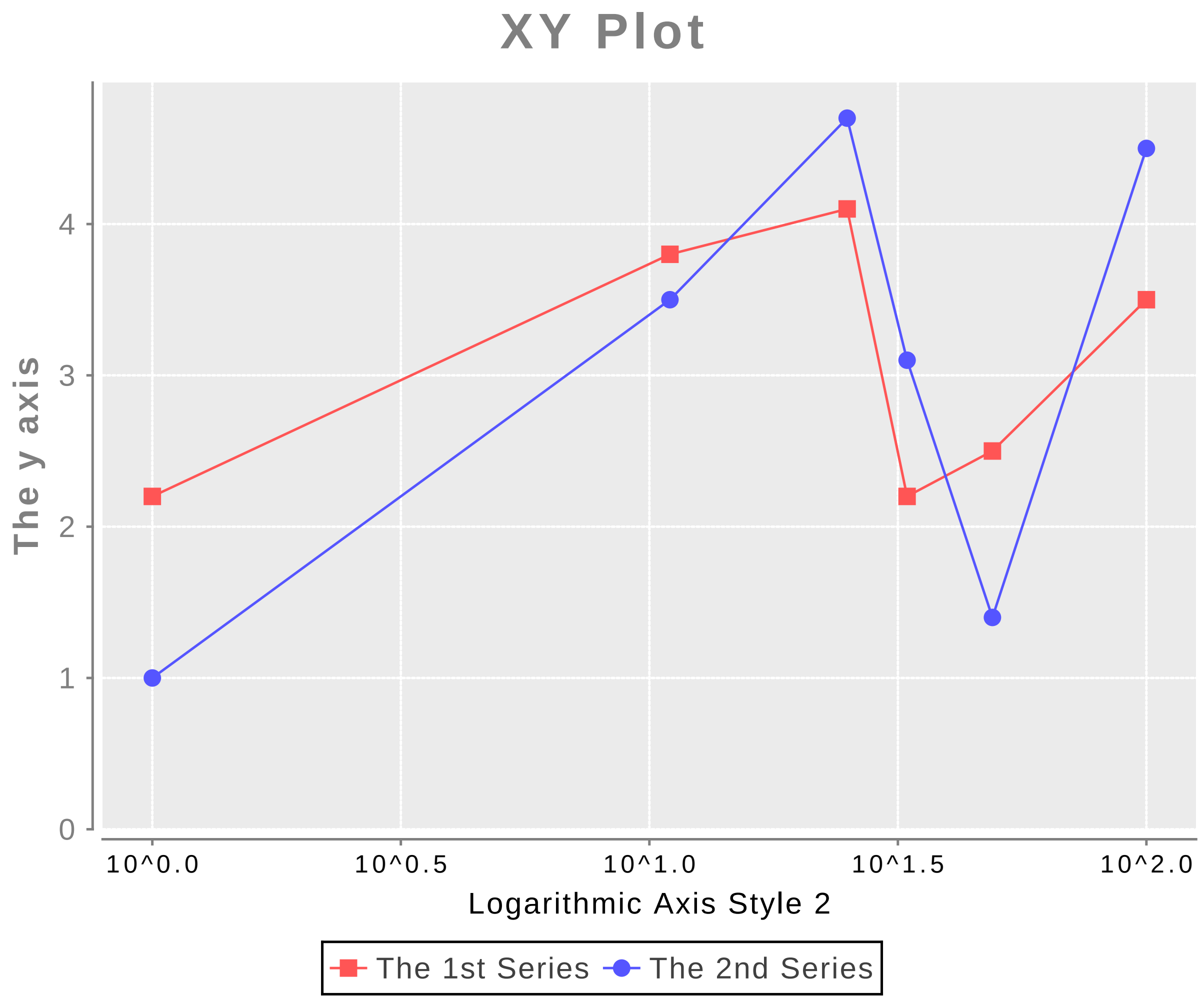

Figure 5: **The chart with the x-axis changed to logarithmic style 1.**

Here is the applicable documentation for the Class introduced in the above code:

http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/LogAxis.html

8 Change the Line Color

This shows how to manually change the default colors used to draw the lines and points.

This code is a little confusing. The setSeriesPaint method is used to set the color. This method acts on a renderer object, thus the renderer is extracted from the plot using the getRenderer method. However, there is a different renderer for each data series. Thus the getRenderer requires an integer parameter indicating which renderer to retrieve. The story is a bit different when invoking the setSeriesPaint method, as the integer index for the series is 0 for both cases.

The Incanter charts set-stroke-color function can be used for this purpose. When there is more than one series of data, the results are different if applied within the scope of a doto or let versus being applied sequentially. When using the Java interop as shown, the results will be the same for both cases.

```
(.setSeriesPaint (.getRenderer plot 0) 0 java.awt.Color/black)
(.setSeriesPaint (.getRenderer plot 1) 0 java.awt.Color/green)
(save-pdf xy1 "./src/jfreechart_tests/xy-line-color1.pdf")
```

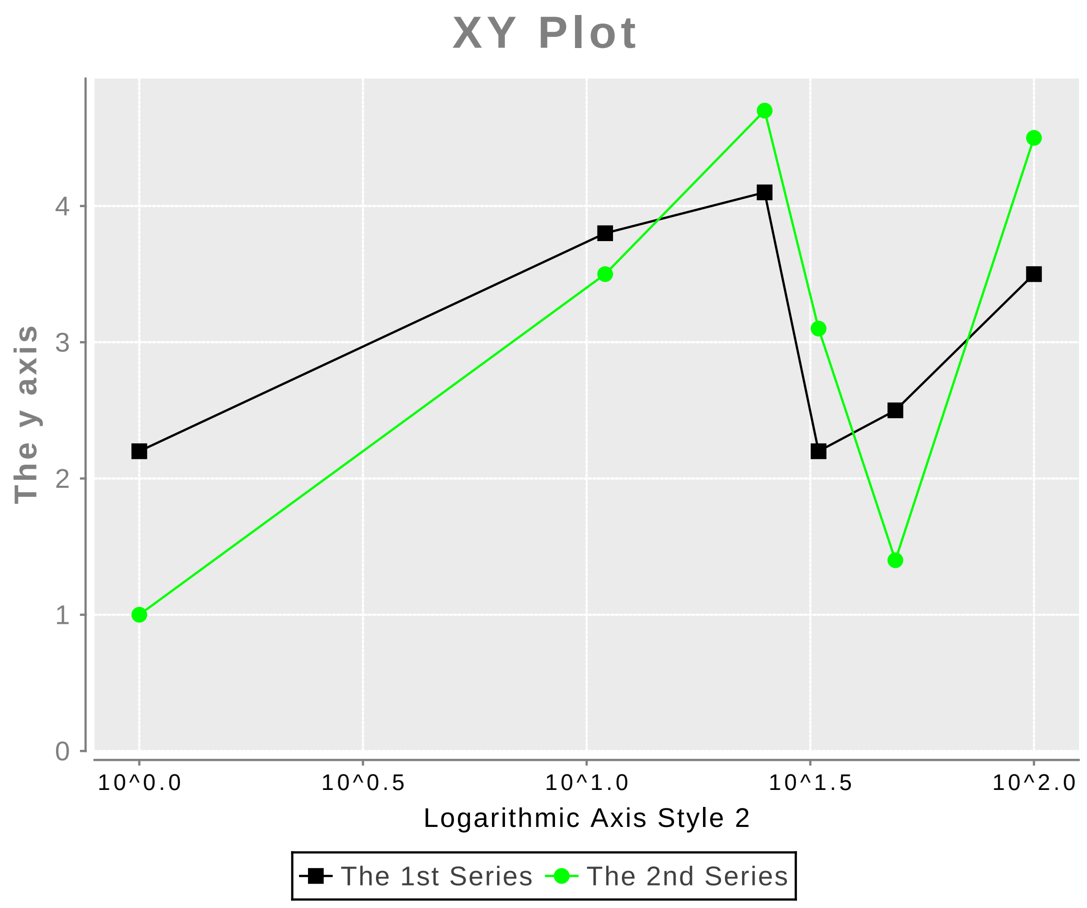

Figure 6: **The lines and points are changed to different colors.**

There are standard color fields, and custom colors are possible:

http://docs.oracle.com/javase/6/docs/api/java/awt/Color.html

9 Change the Stroke

“BasicStroke” is a Java Class which determines the appearance of a drawn line:

http://docs.oracle.com/javase/6/docs/api/java/awt/BasicStroke.html

Note that you may achieve good result with the Incanter charts function set-stroke. As with set-stroke-color, the results achieved may depend on the scope in which it is applied to the chart.

```
(def stroke (java.awt.BasicStroke. 1.0
                                   java.awt.BasicStroke/CAP_ROUND
                                   java.awt.BasicStroke/JOIN_ROUND
                                   1.0
                                   (float-array 1.0 4.0)
                                   0.0))
(.setSeriesStroke (.getRenderer plot) 0 stroke)
(.setSeriesStroke (.getRenderer plot 1) 0 stroke)
(save-pdf xy1 "./src/jfreechart_tests/xy-line-stroke.pdf")
```

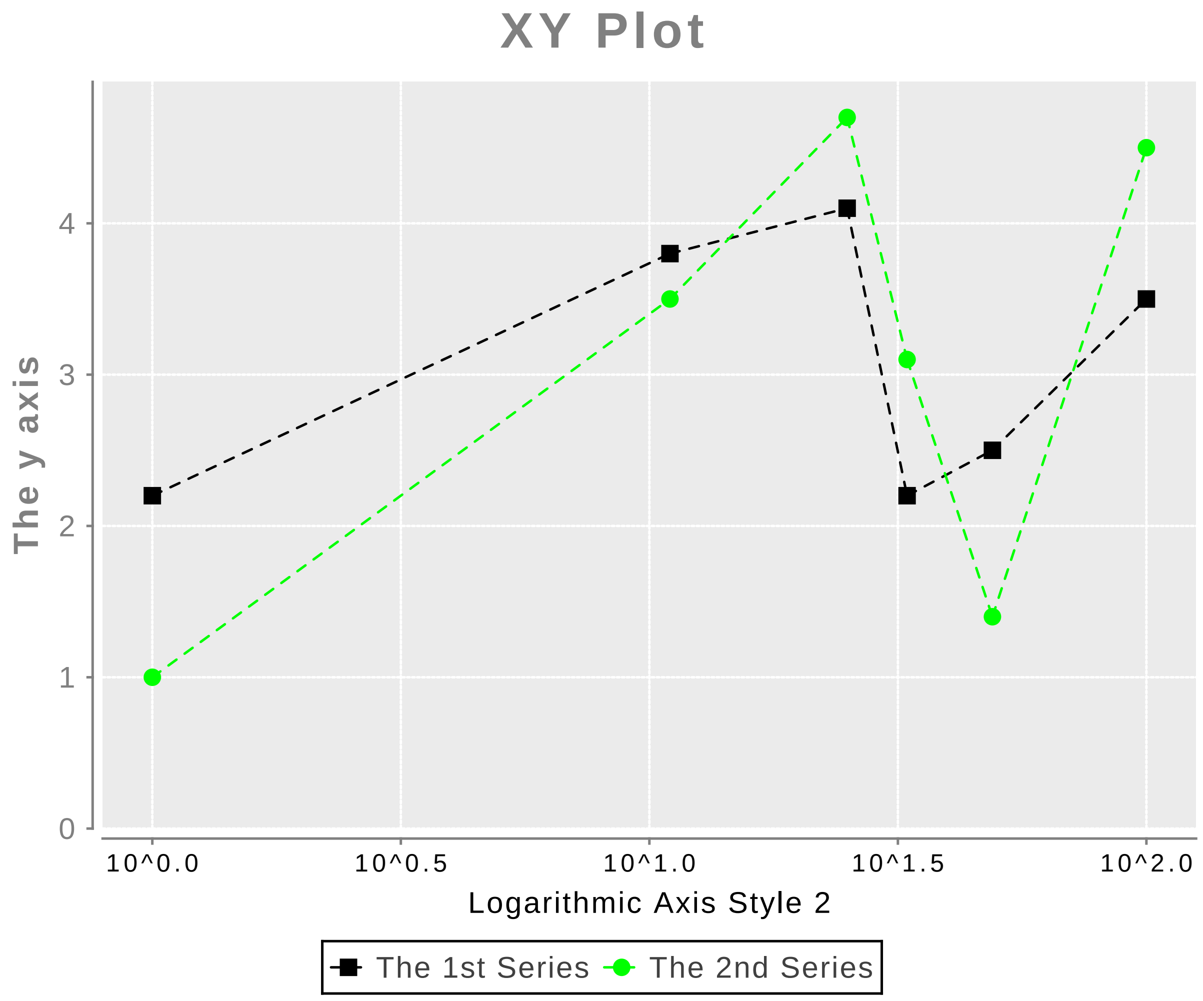

Figure 7: **The “stroke” of the lines is changed to dashed.**

10 A Bar Chart with Whiskers Using Incanter and Clojure

First, set up the namespace and use and import the libraries and JFreeChart classes:

```
(ns jfreechart-tests.core
(:use [incanter core charts pdf]))
(import 'org.jfree.chart.renderer.xy.XYBarRenderer
        'org.jfree.chart.renderer.xy.XYErrorRenderer
        'org.jfree.chart.renderer.GrayPaintScale
        'org.jfree.data.xy.YIntervalSeries
        'org.jfree.data.xy.YIntervalSeriesCollection
        'org.jfree.data.xy.XYBarDataset
        'org.jfree.chart.renderer.xy.StandardXYBarPainter)
```

Create an x-y plot using the Incanter xy-plot function, and save it as a PDF file. View the chart. The errors vector will be used later to add the whiskers.

```
(def xs [0.0 1. 2. 3. 4. 5.])
(def ys [2.2 3.8 4.1 2.2 2.5 3.5])
(def errors [1.0 0.5 0.7 1.1 0.4 0.5])
(def xy1 (xy-plot xs ys :x-label "The x axis" :y-label "The y axis"))
(save-pdf xy1 "./src/jfreechart_tests/xy1-line.pdf")
```

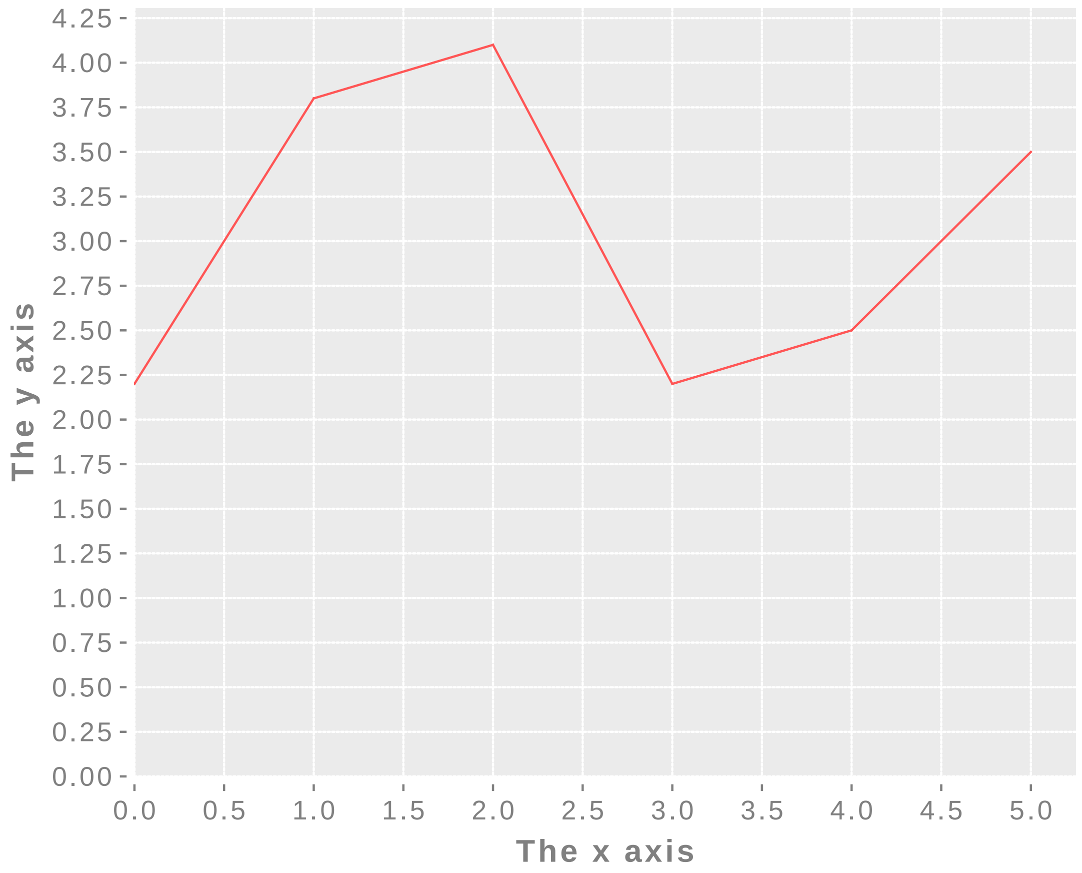

Figure 8: **The basic x-y plot using the Incanter xy-plot function.**

Next, change the xy plot to an "XY Bar Chart" utilizing the Clojure Java Interop and the JFreeChart API. Save the chart and view the changes.

```
;;; First, extract the plot object from the chart object:
(def xy1plot (.getPlot xy1))
;;;  Create an XYBarRenderer and set the margin to 0.2 (this makes a space between the bars).
(def xy1renderer (XYBarRenderer. 0.2))
;;;  Play with some parameters which will affect the appearance of the bars.
(.setShadowVisible xy1renderer false)
;;; Change the color of the bars to gray.  The GrayPaintScale class makes this easier to accompl
(def greypaint (GrayPaintScale. 0 255 100))  ;; 0-255 is the scale, and 100 is the transparency.
(def paint (.getPaint greypaint 150))
(.setSeriesPaint xy1renderer 0 paint)  ;; This makes the bars more like the R version.
;;; Change the default bar painter to the StandardXYBarPainter to get rid of the default gradien
(.setBarPainter xy1renderer (StandardXYBarPainter.))
(.setDrawBarOutline xy1renderer true)
;;; Now change the "renderer" in the xy plot to the XYBarRenderer previously defined.
(.setRenderer xy1plot xy1renderer)
(save-pdf xy1 "./src/jfreechart_tests/xy1bar.pdf")
```

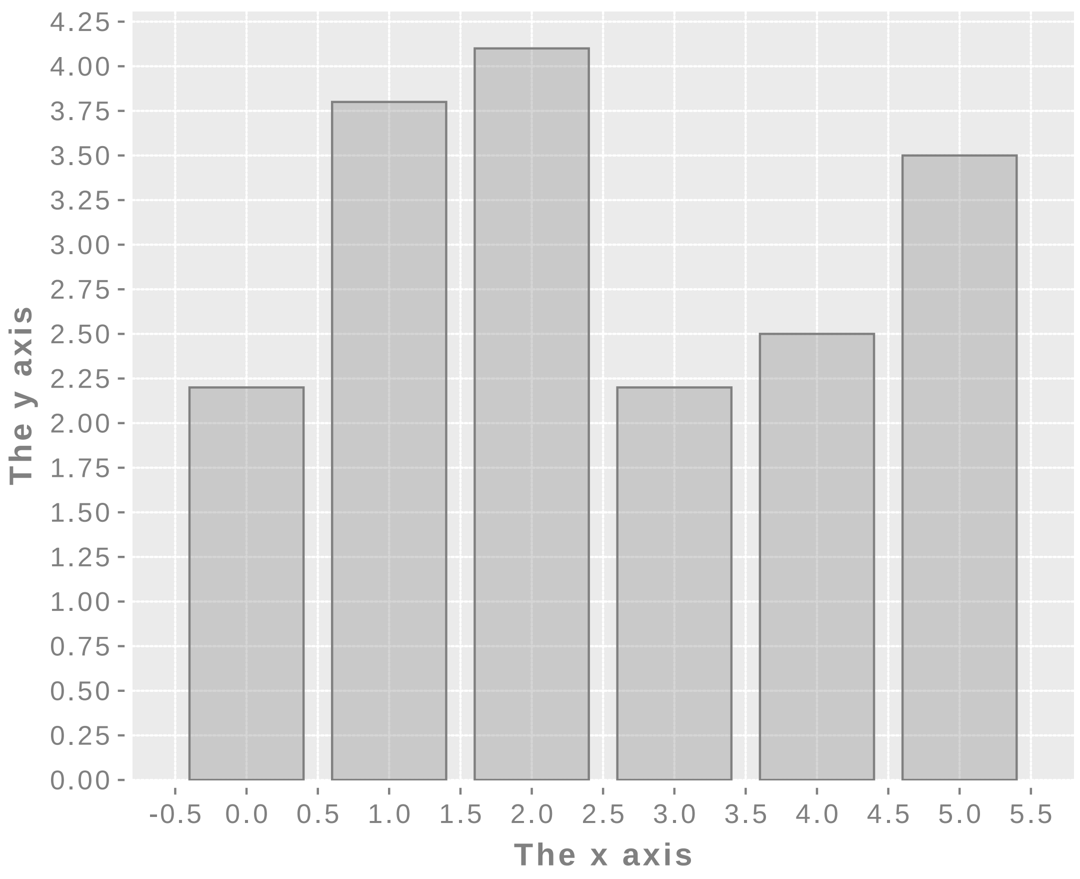

Figure 9: **The basic x-y chart transformed into an x-y bar chart.**

The above code generated an Incanter xy-plot and made several adjustments to the chart's appearance. The most significant of these was changing the renderer to show bars rather than a connected line.

The next step is to add the whiskers. This is done by overlaying the whiskers on top of the already created xy-plot. The key to this is the "XYErrorRenderer" class. A simple function is defined which adds data to the YIntervalSeries.

```
(def xy1errorrenderer (XYErrorRenderer.))
(.setCapLength xy1errorrenderer 30)
(.setRenderer xy1plot 1 xy1errorrenderer)
(def yintervalcollection (YIntervalSeriesCollection.))
(def yintervalseries (YIntervalSeries. "errors"))
(.setSeriesShapesVisible xy1errorrenderer 0 false)
(.setErrorPaint xy1errorrenderer java.awt.Color/black)
;;; A function to add data to a YIntervalSeries.
(defn yintervalseriespop [xs ys errors]
"Populates a YIntervalSeries with data."
(doall (map (fn [a b c] (.add yintervalseries a b (- b c) (+ b c))) xs ys errors)))
(yintervalseriespop xs ys errors)
(.addSeries yintervalcollection yintervalseries)
(.setDataset xy1plot 1 yintervalcollection)
(save-pdf xy1 "./src/jfreechart_tests/xy1bar-whiskers.pdf")
```

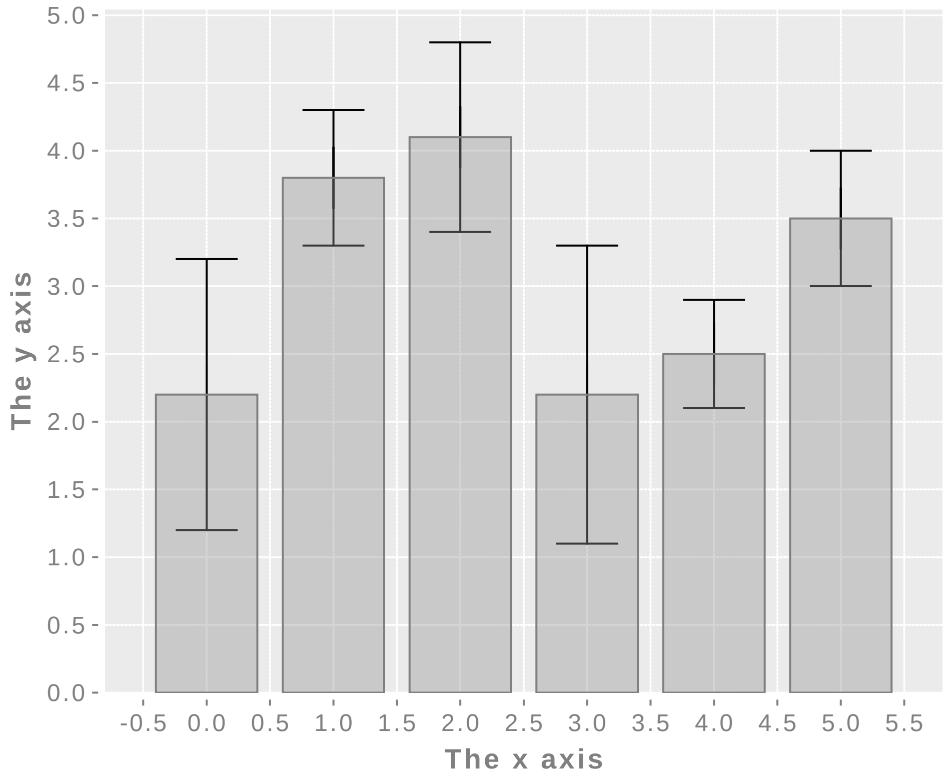

Figure 10: **Whiskers are added to the x-y bar chart.**

11 A Function for Generating an X-Y Bar Chart with Whiskers

Now an attempt to create a function to make an X-Y Bar Chart with the whiskers!

```
(defn xy-bar-chart [xs ys errors x-label y-label]
(let [xychart (xy-plot xs ys :x-label x-label :y-label y-label)
      xyplot (.getPlot xychart)
      xybarrenderer (XYBarRenderer. 0.2)
      _ (.setShadowVisible xybarrenderer false)
      greypaint (GrayPaintScale. 0 255 100)
      paint (.getPaint greypaint 150)
      _ (.setSeriesPaint xybarrenderer 0 paint)
      _ (.setBarPainter xybarrenderer (StandardXYBarPainter.))
      _ (.setDrawBarOutline xybarrenderer true)
      _ (.setRenderer xyplot xybarrenderer)
      xyerrorrenderer (XYErrorRenderer.)
      _ (.setCapLength xyerrorrenderer 30)
      _ (.setSeriesShapesVisible xyerrorrenderer 0 false)
      _ (.setErrorPaint xyerrorrenderer java.awt.Color/black)
      _ (.setRenderer xyplot 1 xyerrorrenderer)
      yintervalcoll (YIntervalSeriesCollection.)
      yintervalser (YIntervalSeries. "errors")
      yintervalserpop (fn [xs ys errors]
        (doall (map (fn [a b c] (.add yintervalser a b (- b c) (+ b c))) xs ys errors)))]
 (yintervalserpop xs ys errors)
 (.addSeries yintervalcoll yintervalser)
 (.setDataset xyplot 1 yintervalcoll) xychart))  ;; End of the function.
;;; Create the chart again using the function:
(def xy-bar-whiskers (xy-bar-chart xs ys errors "The X Axis" "The Y Axis"))
(save-pdf xy-bar-whiskers "./src/jfreechart_tests/xybarwhiskers.pdf")
```

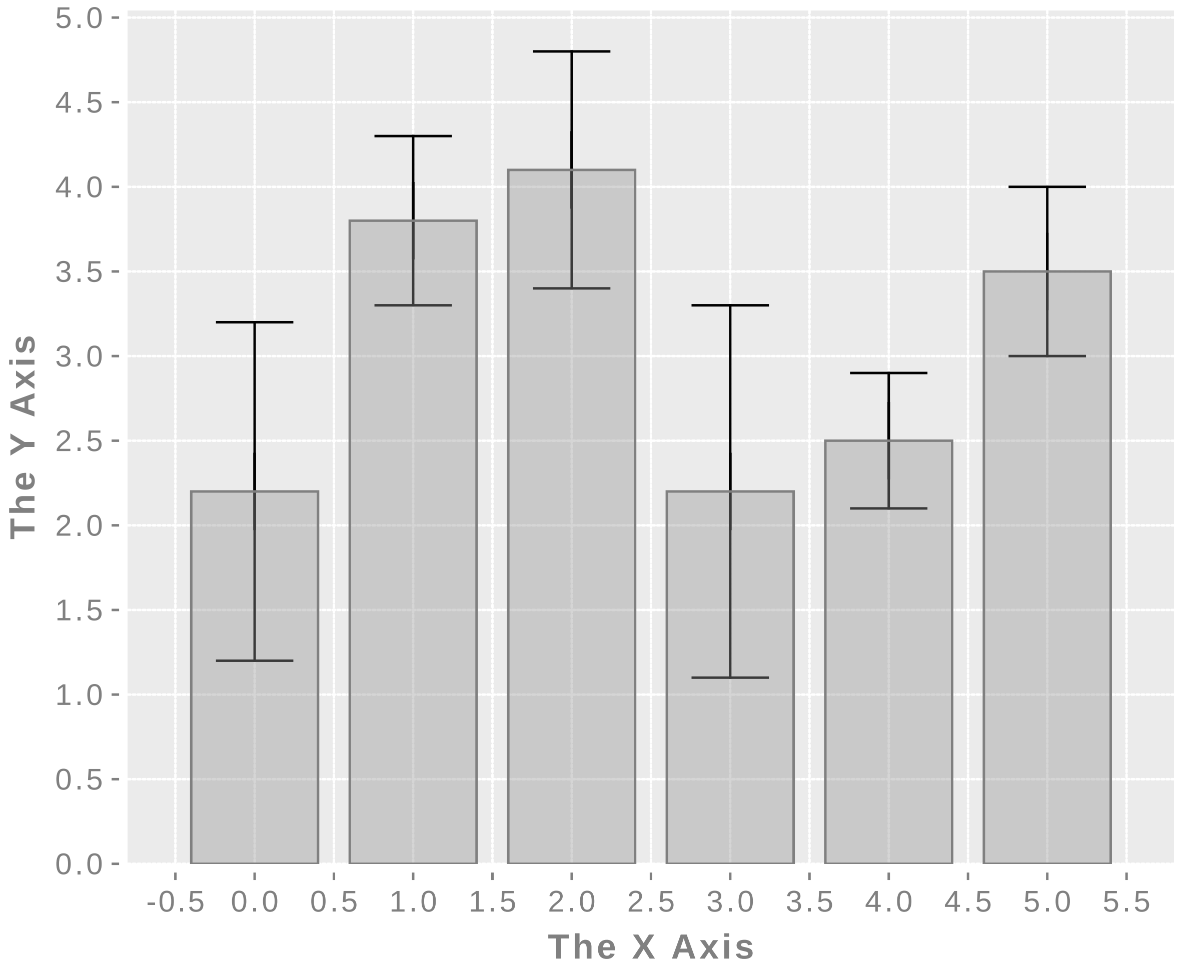

Figure 11: **The same xy-bar-chart generated with a prototype function of the same name.**

12 An XY Plot with Rapid Update

The following is a prototype function to provide immediate updating of an existing chart. The "replace-line" function takes an existing chart and replaces the original data with the data supplied for the x and y arguments.

```
(import     (org.jfree.chart.plot PlotOrientation
                              DatasetRenderingOrder
                              SeriesRenderingOrder)
            (org.jfree.data.xy DefaultHighLowDataset
                             XYSeries
                             XYSeriesCollection)
            (org.jfree.chart.renderer.xy XYLineAndShapeRenderer))
(defn replace-line [chart x y & options]
     (let [opts (when options (apply assoc {} options))
           data (:data opts)
           _x (if (coll? x) (to-list x) ($ x data))
           _y (if (coll? y) (to-list y) ($ y data))
           data-plot (.getPlot chart)
           n (.getDatasetCount data-plot) ;;  This should be 1 for a new chart.
           series-lab (or (:series-label opts) (format "%s, %s" 'x 'y))
           points? (true? (:points opts))
           line-renderer (XYLineAndShapeRenderer. true points?)
           data-set (.getDataset data-plot)  ;; Extract the XYSeriesCollection from the plot.
           ;; Extract the XYSeries from the XYSeriesCollection.
           data-series (.getSeries data-set (.getSeriesKey data-set 0))
           _ (.clear data-series)]  ;; Clear the original data in the series.
       ;; Populate the data series with the supplied data.
       (dorun
        (map (fn [x y]
               (if (and (not (nil? x))
                        (not (nil? y)))
                 (.add data-series (double x) (double y))))
             _x _y))
      (doto data-plot
        (.setSeriesRenderingOrder org.jfree.chart.plot.SeriesRenderingOrder/FORWARD)
        (.setDatasetRenderingOrder org.jfree.chart.plot.DatasetRenderingOrder/FORWARD)
        (.setDataset n data-set)
        (.setRenderer n line-renderer))
      chart))
```

The following code is a simple example showing the usage of the replace-line function.

```
(def chart1 (xy-plot [0 1 2 3 4 5][1 0 3 5 2 1]))
(view chart1)
(replace-line chart1 [0 1 2 3 4 5][1 2 1 4 1 2])
```

13 An XY Plot using Clojure Java Interop (without Incanter)

Here is a code which generates an xy chart using only Java interop to JFreeChart classes. The process goes like this:

- Create an empty XYSeries object.
- Populate the XYSeries object using the "add" method.
- Create an XYSeriesCollection object. The constructor method allows the XYSeries as an argument.
- Create the chart using the static method "createXYLineChart". This returns the chart.

```
(import (org.jfree.chart ChartFactory)
        (org.jfree.data.xy XYSeriesCollection)
        (org.jfree.data.xy XYSeries))
(def x-data [0.0 1.0 2.0 3.0 4.0 5.0])
(def y-data [2.3 9.0 2.6 3.1 8.1 4.5])
(def data-series (XYSeries. 0))  ;; Set the series key to 0.
(dorun  (map (fn [x y]
             (.add data-series (double x) (double y)))  ;; Populate the data series with the sup
             x-data y-data))
(def xy-dataset (XYSeriesCollection. data-series)) ;; Create a dataset and insert the data into
(def xy-chart (org.jfree.chart.ChartFactory/createXYLineChart
                "XY Line Chart"
                "X Axis"
                "Y Axis"
                xy-dataset
                org.jfree.chart.plot.PlotOrientation/VERTICAL
                false
                false
                false))
(save-pdf xy-chart "./src/jfreechart_tests/xy-chart-factory.pdf")
```

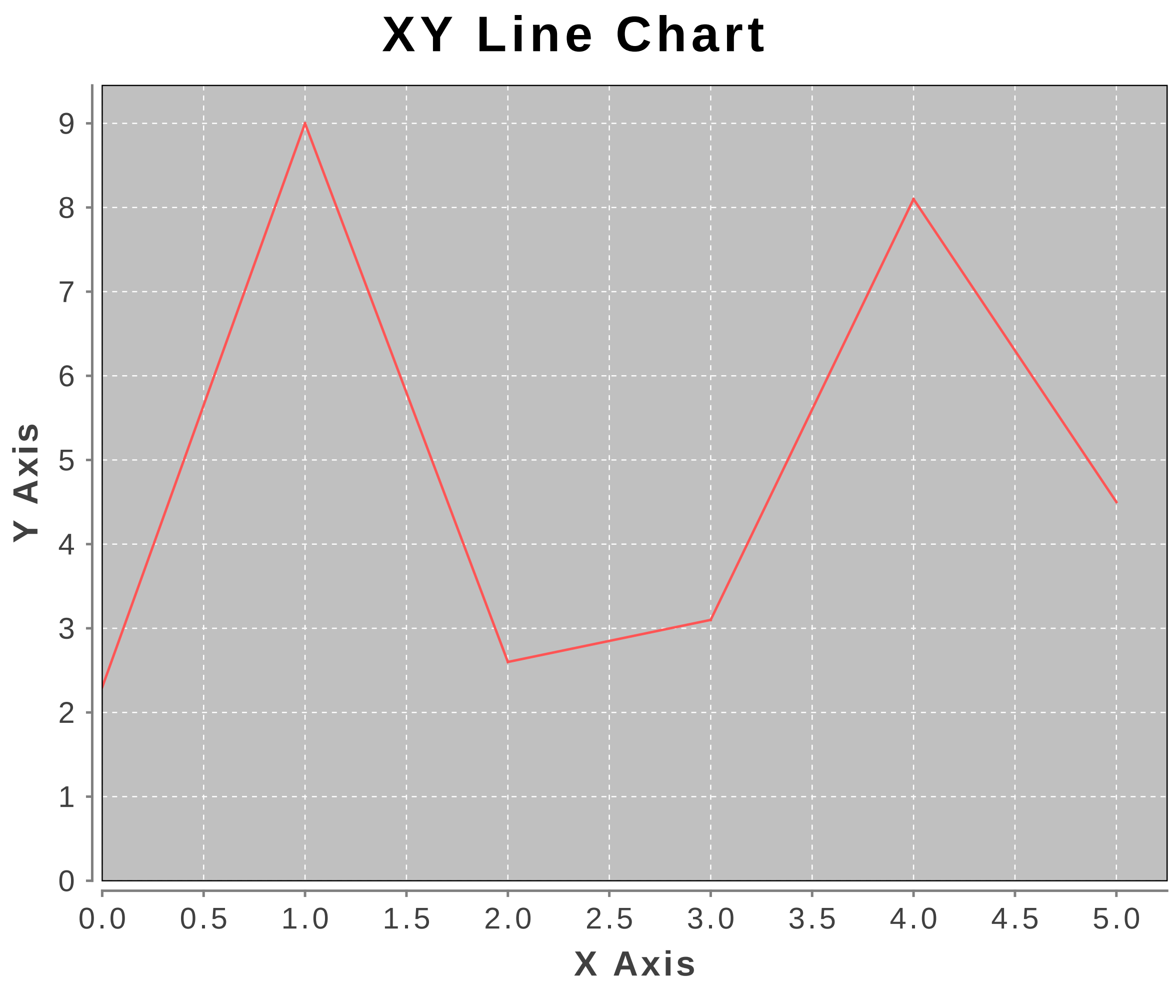

Figure 12: **An xy chart using Java interop to JFreeChart (no Incanter).**

Here are links to the documentation for the JFreeChart classes introduced in the code above:

ChartFactory: "A collection of utility methods for creating some standard charts with JFreeChart."

http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/ChartFactory.html

XYSeriesCollection: this is the "dataset" which stores one or more data series.

http://www.jfree.org/jfreechart/api/javadoc/org/jfree/data/xy/XYSeriesCollection.html

XYSeries: this is the object which stores the numbers.

http://www.jfree.org/jfreechart/api/javadoc/org/jfree/data/xy/XYSeries.html

14 References

Incanter Charts Documentation

http://liebke.github.com/incanter/charts-api.html

Sample plots in Incanter

https://github.com/liebke/incanter/wiki/sample-plots-in-incanter

Clojure Java Interop

http://clojure.org/java_interop

JFreeChart Home

http://www.jfree.org/jfreechart/index.html

JFreeChart API

http://www.jfree.org/jfreechart/api/javadoc/index.html

Clojure API

http://clojure.github.com/clojure/