# Advanced AIDA Tutorial

⚠️ THIS PAGE IS UNDER CONSTRUCTION

## AIDA for Power Users



### Introduction

This Document covers advanced topics in JAS3 and AIDA. It assumes that you are familiar with AIDA and that you have completed the Getting Started Tutorial
The examples in this section are mostly written in the Python language. It's syntax is simpler and less strict than Java, yet very clear. If you do not feel comfortable learning yet another language, please visit the exellent, if slightly dated AIDA documentation. However, please feel encouraged to take a look at the examples. They should all run out of the box (if not, please don't feel discouraged – file a bug report), so that you should still be able to get something out of them.

### Secondary Y Axis

Comparing two datasets in the same plot is a very convenient feature in AIDA. However, those two datasets may not always have the same scale, or even the same units. JAS3 lets you add a secondary Y Axis, which can be scaled independently of the primary one. Just select the Data tab in the `Plot Properties` dialog. For each data set listed on the left-hand side, you can select which axis to associate it with by selecting either Y0 for the primary Y Axis on the left-hand side of the plot, or Y1 for the secondary Y Axis on the right-hand side of the plot. The properties of each axis, like range and label can be changed in the Y Axis tab.
You can also do it in a program:

```
import hep.aida.*;
import java.util.Random;

public class SecondAxis {
    public static void main(String[] argv) {

        IAnalysisFactory af = IAnalysisFactory.create();
        ITree tree = af.createTreeFactory().create();
        IHistogramFactory hf = af.createHistogramFactory(tree);

        IHistogram1D h1 = hf.createHistogram1D("h1",100,-6,4);
        IHistogram1D h2 = hf.createHistogram1D("h2",100,-4,6);

        Random r = new Random();

        for ( int i = 0; i<1000; i++ ) {
            h1.fill(r.nextGaussian()-1.);
            h2.fill(r.nextGaussian()+1.);
            h2.fill(r.nextGaussian()+1.);
        }

        IPlotterFactory plotterFactory = af.createPlotterFactory();
        IPlotter plotter = plotterFactory.create("Styles.java plot");

       // Change the default styles for the region to hide
       // legend and statistics
        IPlotterStyle regionStyle = plotter.region(0).style();
        regionStyle.legendBoxStyle().setVisible(false);
        regionStyle.statisticsBoxStyle().setVisible(false);


        // >>>>> This style overwrites default "Y0" position of y axis
        //
        IPlotterStyle style = plotterFactory.createPlotterStyle();
        style.yAxisStyle().setParameter("yAxis", "Y1");
        //
        // >>>>>

        plotter.region(0).plot(h1);
        plotter.region(0).plot(h2, style);

       plotter.show();
    }
}
```

## Changing Layout/Style/Properties with a Script

While it is convenient to be able to manipulate your data with the mouse, in many cases you will want to apply the same modifications to a set of similar data, or perform more complicated modifications that are quite possible to do with AIDA, but not (yet) implemented in JAS3. This is where a script comes in handy

### Advanced Manipulations of Histograms

## Adding and subtracting histograms

## Fitting

### Using predefined functions

### Writing your own function

Oftentimes you will have to perform more complicated computations than can be put in a string. Use this template to

```
from hep.aida import IFunction
from org.lcsim.util.aida import AIDA
import math

class MyFunc(IFunction):
    def __init__(self, track):
        self.title = "My Own Function"
        return

    def providesGradient(self):
        return false

# This Function returns the value f(x)
# x can be multi-dimensional
    def value(self, x):
        y = x[0] + math.exp(x[1]) + x[0]*x[1]
        externalValue = someExternalFunction(y, x)
        return externalValue-y

# This tells AIDA the dimension of the function
    def dimension(self):
        return 2

    def annotation(self):
        ann = IAnnotation()
        ann.addItem('Title', self.title)
        return ann

def someExternalFunction(y, x):
    return 2*y+3*x[0]

def main():
    tree = aidaMasterTree
    aida = AIDA.defaultInstance()
    funcFac = aida.analysisFactory().createFunctionFactory(tree)
```

## Tuples

### Flat Tuples

### Nested Tuples