ENTHOUGHT
SCIENTIFIC COMPUTING SOLUTIONS

# Data Visualization with Python

Pankaj Pandey
&
Puneeth Chaganti

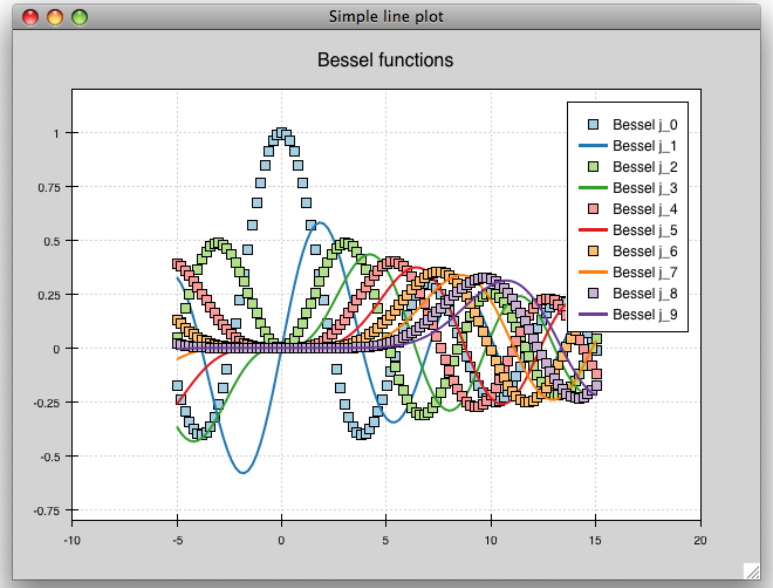
# Demos

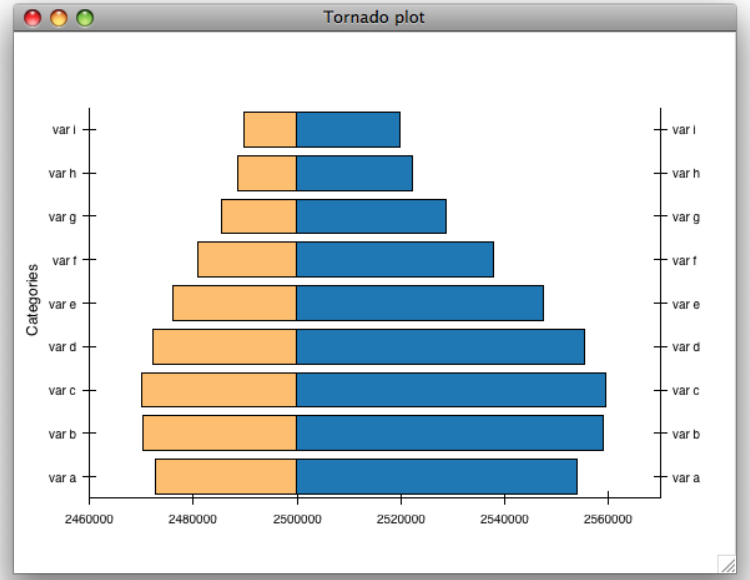
# Talk outline

- What and Why
- First-plot
  - Step by step code explanation
  - A short detour into Traits & TraitsUI
- An overview of Chaco's architecture
  - Commonly used models and classes
- More examples

# What is Chaco?

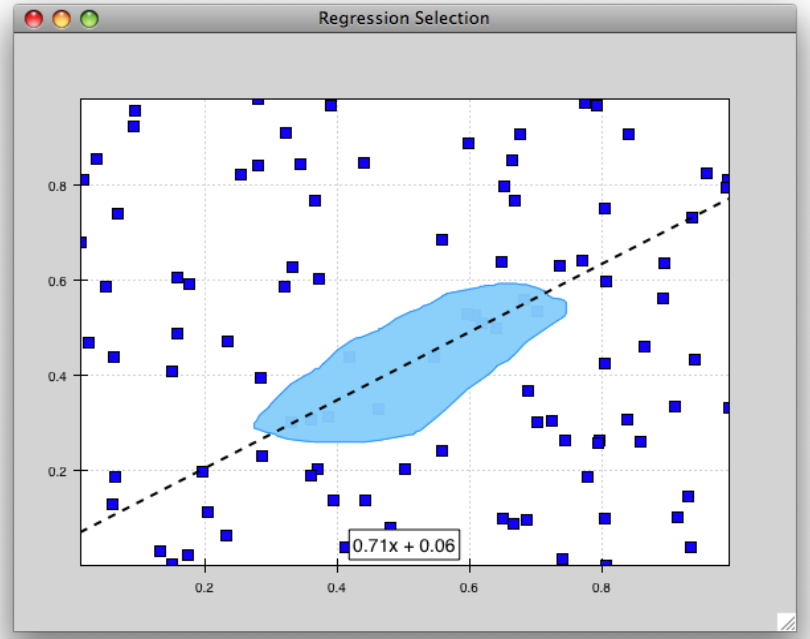
- A *plotting application toolkit* for Python
- You can build simple, static plots
- But, really shines in **interactive** plotting

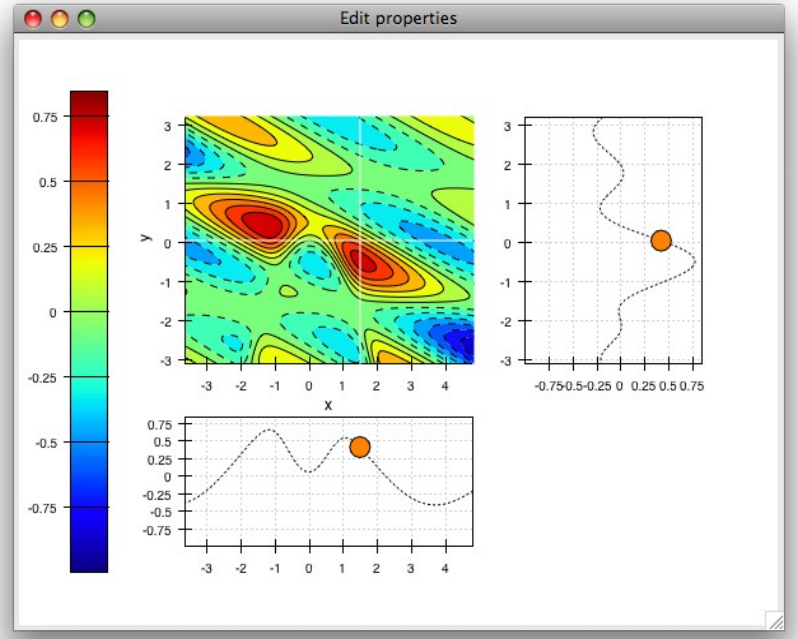


# Why Chaco?

- Plots update live as data changes
- Linked plots
- Extensible architecture
- GUI toolkit agnostic



# Shell mode

```
from numpy import *
from chaco.shell import *

x = linspace(-2*pi, 2*pi, 100)
y = sin(x)

plot(x, y, 'r-')
title('First plot')
ytitle('sin(x)')
show()
```

# First Plot

```
class LinePlot(HasTraits):

    plot = Instance(Plot)
    traits_view = View(
        Item('plot',editor=ComponentEditor(), show_label=False),
        width=500, height=500,
        resizable=True,
        title="Chaco Plot")

    def _plot_default (self):
        x = linspace(-14, 14, 100)
        y = sin(x) * x**3
        plotdata = ArrayPlotData(x = x, y = y)

        plot = Plot(plotdata)
        plot.plot(("x", "y"), type="line", color="blue")
        plot.title = "sin(x) * x^3"
        return plot


if __name__ == "__main__":
    LinePlot().configure_traits()
```

# Traits & TraitsUI

- Standardization
  - Initialization
  - Validation
  - Delegation
- Visualization  (TraitsUI)
- Notification
- Documentation

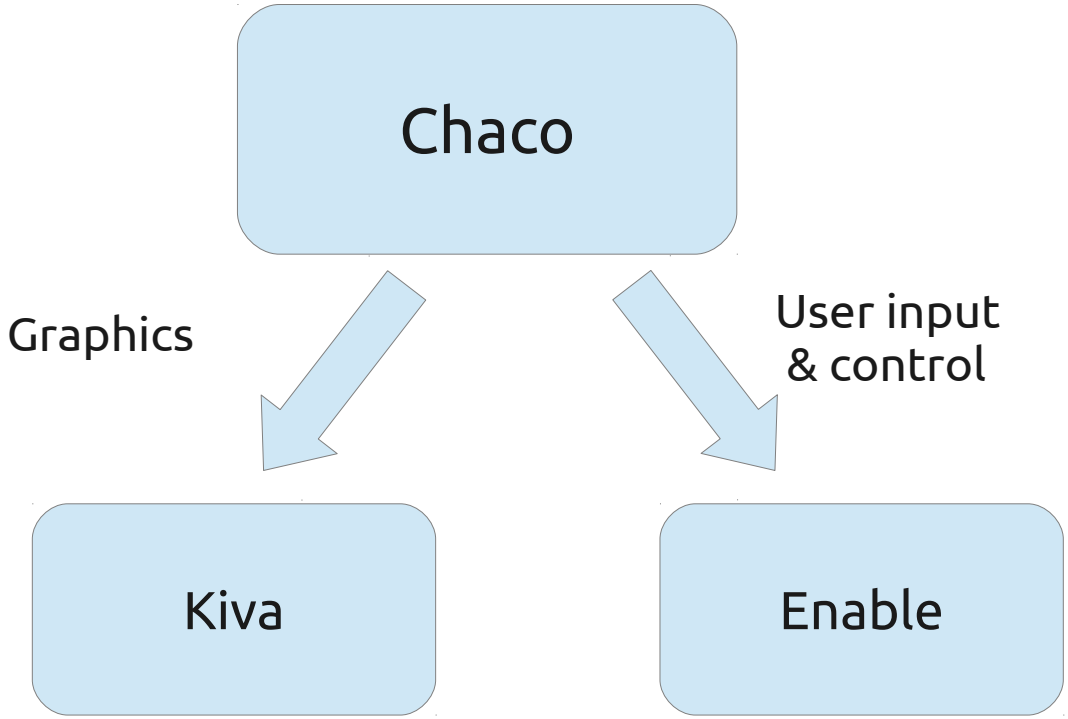
# Architecture Overview

## Core Ideas

- Plots are compositions of visual components
- Separation between data and screen space
- Modular design and extensible classes

ENTHOUGHT
SCIENTIFIC COMPUTING SOLUTIONS

# Architecture Overview



ENTHOUGHT
SCIENTIFIC COMPUTING SOLUTIONS

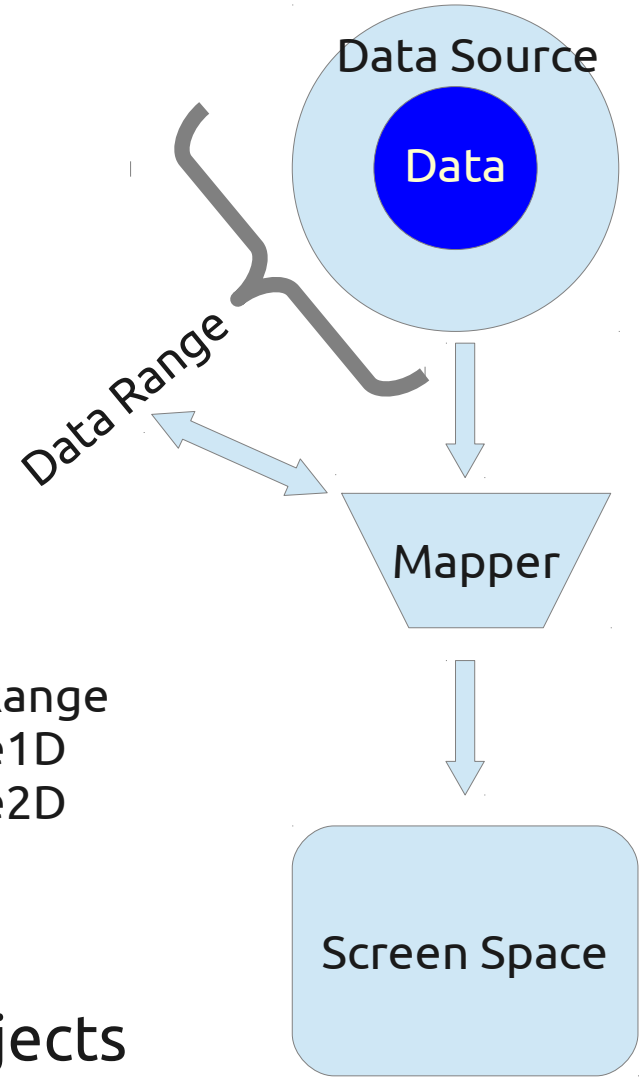
# Architecture Overview

From 20000 ft., Chaco consists of –

- Data handling classes – wrap input data, interface with application specific data sources, transform co-ordinates between data and screen space (eg., ArrayDataSource, LinearMapper)
- Visual components – render to the screen (eg. LinePlot, ScatterPlot, Legend, PlotAxis, …)
- Tools – handle keyboard or mouse events and modify other components (eg., PanTool, ZoomTool, ScatterInspector)

# Architecture Overview

## Commonly used Modules and Classes




- ArrayDataSource
- DataContextDataSource
- GridDataSource
- ImageData
- MultiArrayDataSource
- PointDataSource
  - data_changed
  - bounds_changed
  - metadata_changed

- BaseDataRange
- DataRange1D
- DataRange2D

- Base1DMapper
- LinearMapper
- LogMapper
- GridMapper
- PolarMapper

Data Objects

ENTHOUGHT
SCIENTIFIC COMPUTING SOLUTIONS

# Architecture Overview

## Commonly used Modules and Classes

### Containers

- Handle layout
- Similar to layout grids in GUI toolkits
- Efficient way for event dispatch, since screen space is partitioned logically
  - OverlayPlotContainer
  - HplotContainer
  - VplotContainer
  - GridPlotContainer

### Tools

- Take events from a component and perform actions based on that
  - PanTool
  - ZoomTool
  - ...

### Renderers

- Actually draw a type of plot
  - BarPlot
  - Base2DPlot
  - ContourLinePlot
  - ContourPolyPlot
  - ImagePlot
  - CMapImagePlot
  - LinePlot
  - ErrorBarPlot
  - PolygonPlot
  - FilledLinePlot
  - ScatterPlot
  - ColormappedScatterPlot
  - ColorBar
  - PolarLineRenderer

### Overlays

# Thank You!