

Data Visualisation and Graphics

Rich Park



APL is great for:

Data manipulation & pre-processing

- Cleaning $\alpha \neq \omega$ $\alpha \square \omega$ $\alpha \uparrow \omega$ $\alpha \downarrow \omega$
- Reformatting ϕ ϕ ϕ $\square \text{CSV}$ $\square \text{JSON}$

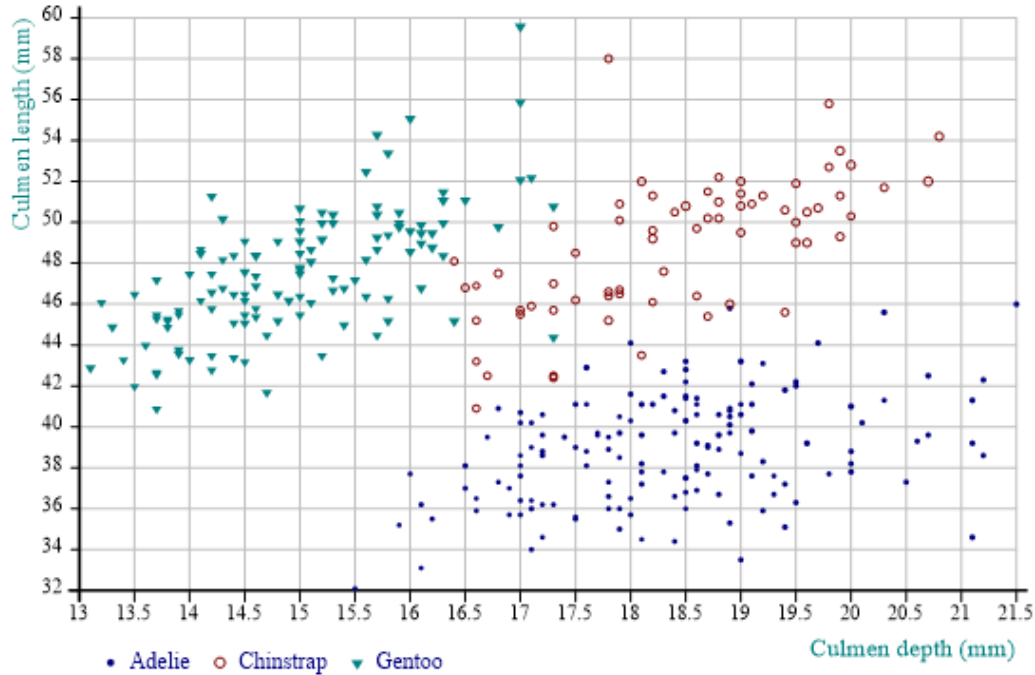
Exploration

- Summaries: $+ / \omega$ \times / ω $\lceil /$ $\lfloor /$
- Grouping: $F \boxtimes \omega$
- Sort/Lookup: $\uparrow \omega$ $\alpha \in \omega$ $\alpha \iota \omega$



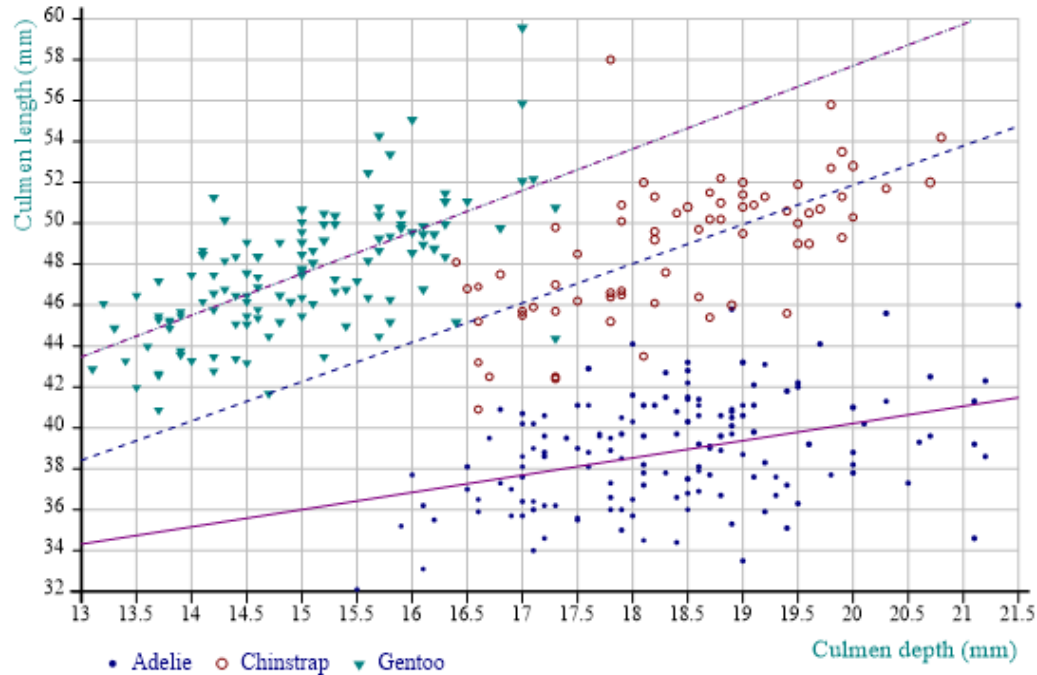
Communicating ideas

Antarctic Penguin Culmen Size and Penguin Mass



Discovering patterns

Antarctic Penguin Culmen Size and Penguin Mass



Coming Up...

- SharpPlot/Vega
- HTMLRenderer/□WC/Observable
- DUI/Abacus
- A little bit of speculation



Previously...

2013: The Future of SharpPlot and RainPro:

Nicolas Delcros dyalog.tv/Dyalog13/?v=Xo3vRQMaPxo

2018: Cross-Platform Charting:

Brian Becker dyalog.tv/Webinar/?v=wapJgEXSzvc

2020: APL Cultivations (Orchard chat lessons) 44 and 45

Nicolas Delcros apl.wiki/APL_Cultivation



SharpPlot

- ✧ Cross platform
- ✧ Code examples from C#
- ✧ Argument nesting is fiddly
- ✧ State machine

```
'InitCauseway' □CY 'sharpplot' ◇ InitCauseway0
```

```
sp ← □NEW Causeway.SharpPlot
```

```
    ↪ sp.Set<things>  
    sp.Draw<thing> <data
```

```
sp.RenderSvg Causeway.SvgMode.FixedAspect
```



Map Class

Leaf Reference

Leaf Class

Leaf Properties

Leaf Methods

Class

Properties

Methods

Leaf Structures

Type Structure

GraphStyle Structure

Style Structure

Leaf Enumerations

Layout Classes

PageElement Class

PageBlock : PageElement Class

PageBlock : PageElement Class

PageElement Class

PageElement Class

Page Reference

Page Class

PageMath Class

Page Class

Page

Page Notes

Page

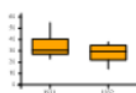
Page

DrawBarChart



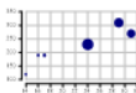
Draws a grouped or stacked barchart from an array of arrays of values

DrawBoxPlot



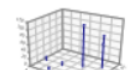
Construct compound box-whisker plot from a data array and two integer category arrays

DrawBubbleChart



Scatterplot from 2 arrays of data with markers scaled by 3rd array

DrawCloudChart



3D Scatter from x,y and multiple z values

sharpplot.com/SharpPlot-Methods.htm

DrawContourPlot



DrawDialChart



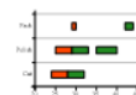
Draw a dial or gauge from data and pointer radii

DrawFrame



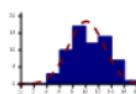
Draw current frame, with any fixed text such as headings and footnotes

DrawGanttChart



Gantt chart from 3 double arrays of Y-values, X-start and X-end values

DrawHistogram



Construct a histogram from a floating-point vector



Cross-platform charts in Dyalog APL

The best way to create cross-platform charts in Dyalog APL is to use the **InitCauseway** function. It will pick the .Net assembly if it's available, else fall back to pure-APL SharpPlot.

It will initialise the argument namespace (current space if θ is given), by creating a **Causeway** and a **System.Drawing** namespace there, so that code can be using interoperably on all platforms supported by Dyalog. The advantage over the previous strategies is that the namespace will be cluttered with fewer names (only Causeway, System and Drawing, rather than the whole list of structures used by Causeway).

However all structures will have to be fully qualified : structures from the Causeway assembly (all in this documentation) will have to be prepended with **Causeway**, and System.Drawing structures (Color, FontStyle) will have to be prepended with **System.Drawing**. That way the script will be completely cross-platform.

All you need to do to initialise the namespace once with:

```
'InitCauseway' 'view' ⌈CY 'sharpplot'  
InitCauseway  $\theta$  ⌈A initialise current namespace
```

The

sharpplot.com/Languages.htm

```
data←33 18 12 10
```

```
exp←0 0 0 20
```

```
key←'UK' 'France' 'Italy' 'Ireland'
```

```
sp←⌈NEW Causeway.SharpPlot
```

⌈A default size

```
sp.Heading←'Sample pie chart'
```

⌈A Property

```
sp.PieChartStyle←Causeway.PieChartStyles.ValueTags
```

⌈A Flag property with single flag

```
sp.KeyStyle←Causeway.KeyStyles.(CenterAlign+BottomAlign)
```

⌈A Flag property with multiple flags

```
sp.SetKeyText(=key)
```

⌈A Need to enclose the key vector because it is a single argument, rather than a list of arguments

```
sp.SetColors ⌈System.Drawing.Color.(Navy Maroon Teal Green)
```

⌈A Need to enclose the vector because it is a single argument

```
sp.SetMargins(42 48 36 18)
```

⌈A No need to enclose because the 4-item vector maps to 4 arguments

```
sp.DrawPieChart(data exp)
```

⌈A Two arguments

```
sp.SaveSvg(=samplepie.svg')
```

⌈A SharpPlot/APL doesn't support raster graphics : SVG is the recommended way to export charts.

```
svg←sp.RenderSvg  $\theta$ 
```

⌈A Return SVG image source

▽



- # Data Visualisation



Vega

- Define using JSON
 - Hand write in an editor
 - Programmatically generate using JSON



Vega

```
{
  "config": {
    "axis": true
  },
  "data": {
    "values": [
      {"count": 161, "item": "5"},
      {"count": 166, "item": "3"},
      {"count": 166, "item": "2"},
      {"count": 157, "item": "1"},
      {"count": 171, "item": "6"},
      {"count": 179, "item": "4"}
    ]
  },
  "encoding": {
    "x": { "field": "item", "type": "nominal" },
    "y": { "field": "count", "type": "quantitative" }
  },
  "height": 300, "mark": "bar", "width": 500
}
```

g JSON

e i
na

```
vega←[]JSON'{'
vega.config←[]JSON'{'
vega.config.axis←'true'
data←[]JSON'{'
V2J←{j←[]JSON'{' ⋄ j.(item count)←α ω ⋄ j}
data.values←[]JSON'{'
data.values←V2J/ω
vega.data←data
vega.mark←'bar'
vega.encoding←[]JSON'{'
x_enc←'{"field": "item", "type": "nominal"}'
y_enc←'{"field": "count", "type": "quantitative"}'
vega.encoding.x←[]JSON x_enc
vega.encoding.y←[]JSON y_enc
vega.width←500
vega.height←300
( []JSON[]'Compact' 0)vega
```



Nic



I guess the interesting bit is how to embed charts in <whatever you're trying to do>

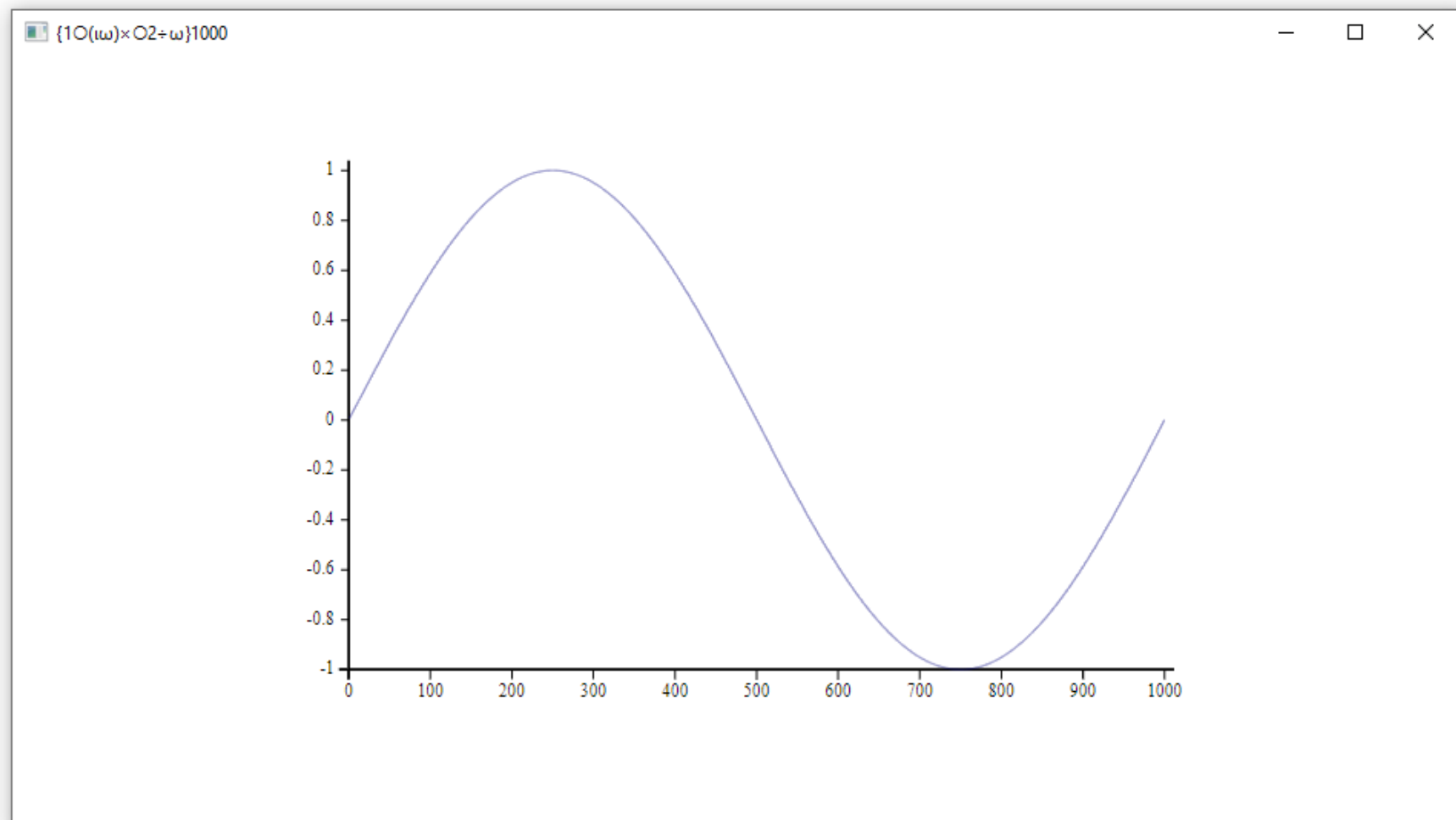


Publishing and embedding

- ✧ Print (PDF)
 - ✧ Raster (PNG, BMP, JPG)
 - ✧ **SVG**
 - ✧ XAML
-
- ✧ \square WC
 - ✧ HTMLRenderer
 - ✧ Browser-based (served)



```
]plot {10(iω)×02÷ω}1000
```



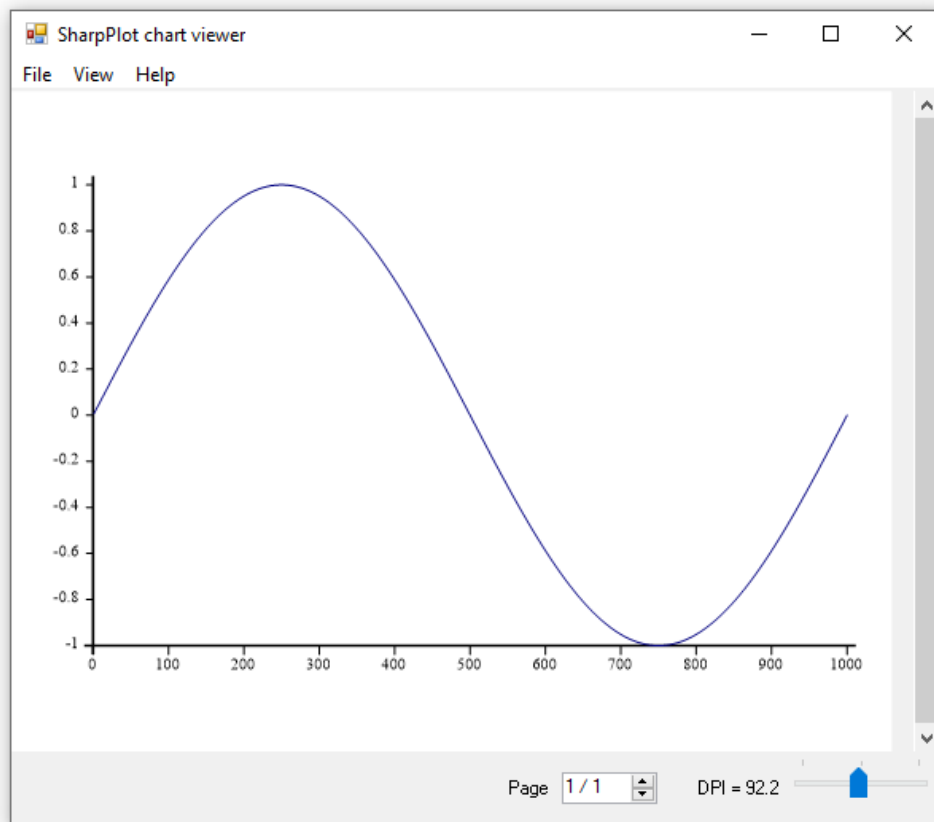
.NET SharpPlot Viewer




```

'InitCauseway' 'View' CY 'sharpplot' ♦ InitCauseway θ
sp←NEW Causeway.SharpPlot
.NET sp.DrawLineGraph<{10(ιω)×o2÷ω}1000
View sp

```



Embed in Windows GUI



SharpPlot graph into a Windows Form

POSTREPLY ↩

Search this topic...

Search

18 posts • Page 2 of 2 • 1 2

RE: SHARPPLOT GRAPH INTO A WINDOWS FORM

by [Tomas Gustafsson](#) on Sun Jan 30, 2022 8:27 pm

Beautiful tones, thank you Pierre! I was unaware of the ElementHost control.

In fact I think short samples of both might be valuable for the community? For me personally, `WC` is the preferred one for now. If you are willing to spend a few moments on this, I'd be most grateful as might many others too!

Tomas Gustafsson

Posts: 100

Joined: Mon Sep 19, 2011 6:43 pm

RE: SHARPPLOT GRAPH INTO A WINDOWS FORM

by [PGilbert](#) on Mon Jan 31, 2022 2:14 am

Here are 2 ways to show the Xaml obtained.

Get a WPF Viewbox from the Xaml representation of the Graph:

CODE: [SELECT ALL](#)

```
#.NET.sp1.DrawLineGraph(,c110)(110)

A Render the above Graph as a Xaml text file with 96 dpi
xamlPlot<#.NET.sp1.RenderXaml 96

A Get the WPF Canvas object from the Xaml.
#.NET.␣USING␣<'System.Windows.Markup,WPF/PresentationFramework.dll'
#.NET.canvasObj<#.NET.XamlReader.Parse(=xamlPlot)

A Get a WPF Viewbox object with the Canvas object as its child.
#.NET.␣USING␣<'System.Windows.Controls,WPF/PresentationFramework.dll'
```



PGilbert

Posts: 434

Joined: Sun Dec 13, 2009 8:46 pm

Location: Montréal, Québec, Canada



HTMLRenderer

- ✧ Cross platform
- ✧ Renders SVG
- ✧ Use web stack (HTML/CSS/JavaScript)
- ✧ But I don't want to write JS?



Dyalog User Interface (DUI)



Dyalog User Interface (DUI)

MiServer is a free, open-source web server implemented in Dyalog APL. It enables the APL user to build sophisticated websites with little to no knowledge of the web technologies like HTML, JavaScript, etc. We hope that you will enjoy MiServer; join the [forums](#), and contribute via [GitHub](#).

This website serves as guide to MiServer 3 – providing documentation, samples, and advice. At the same time, it showcases many of MiServer's capabilities. If you are just getting started with MiServer 3, check out the Tutorial tab. Look through the Samples to get an idea of the variety of web content that MiServer can generate.

Explore the tabs for documentation, the broad selection of available controls, and the vast collection of sample pages and mini-apps. See the source of any page by clicking the MS3 logo in the top left corner and access other resources by clicking the Resources link in the top right corner.



Creating a MiSite
Creating a Widget
Creating Content
Event Handling
Testing with Selenium
Install and Config
└ **Configuration**
└ **Installation**
References
└ **Function Reference**
└ **Terminology**



Carlisle Abacus

github.com/the-carlisle-group/Abacus

- ✧ Manipulate a Document Object Model (DOM) directly in APL
- ✧ Convert APLDOM $\leftarrow \rightarrow$ HTML
- ✧ Serve in HTMLRenderer



Observable

- ✧ Collaborative data story telling
- ✧ Easy to insert basic plots
- ✧ Compatible with Vega/D3



Data Visualisation

Edward Tufte

Keep it simple – less clutter

Present in a digestible format

Jacques Bertin

Semiology of Graphics

Mapping of visual properties to data relations

Lee Wilkinson

Grammar of Graphics

Inspiration: ggplot2 (R)

D3 (JavaScript)



Present issues, future possibilities

- ✧ Read lots of docs
- ✧ Learn something that "isn't APL"
- ✧ Translate code examples
- ✧ Alter your data to fit the framework (easy)



Present issues, future possibilities

- ✧ WIBNI we had an APL graphic manipulation framework a la D3/ggplot2?
- ✧ Abacus+ ?
- ✧ Something else?



APL Media

britishaplassociation.org/webinar-schedule-2022

24th Feb – BAA: Open Session

Dyalog.tv

17th March – Webinar: Link & Sharing Code

(~30th March) – APL Seeds '22

