TECHNISCHE UNIVERSITÄT
KAISERSLAUTERN

TRR
175

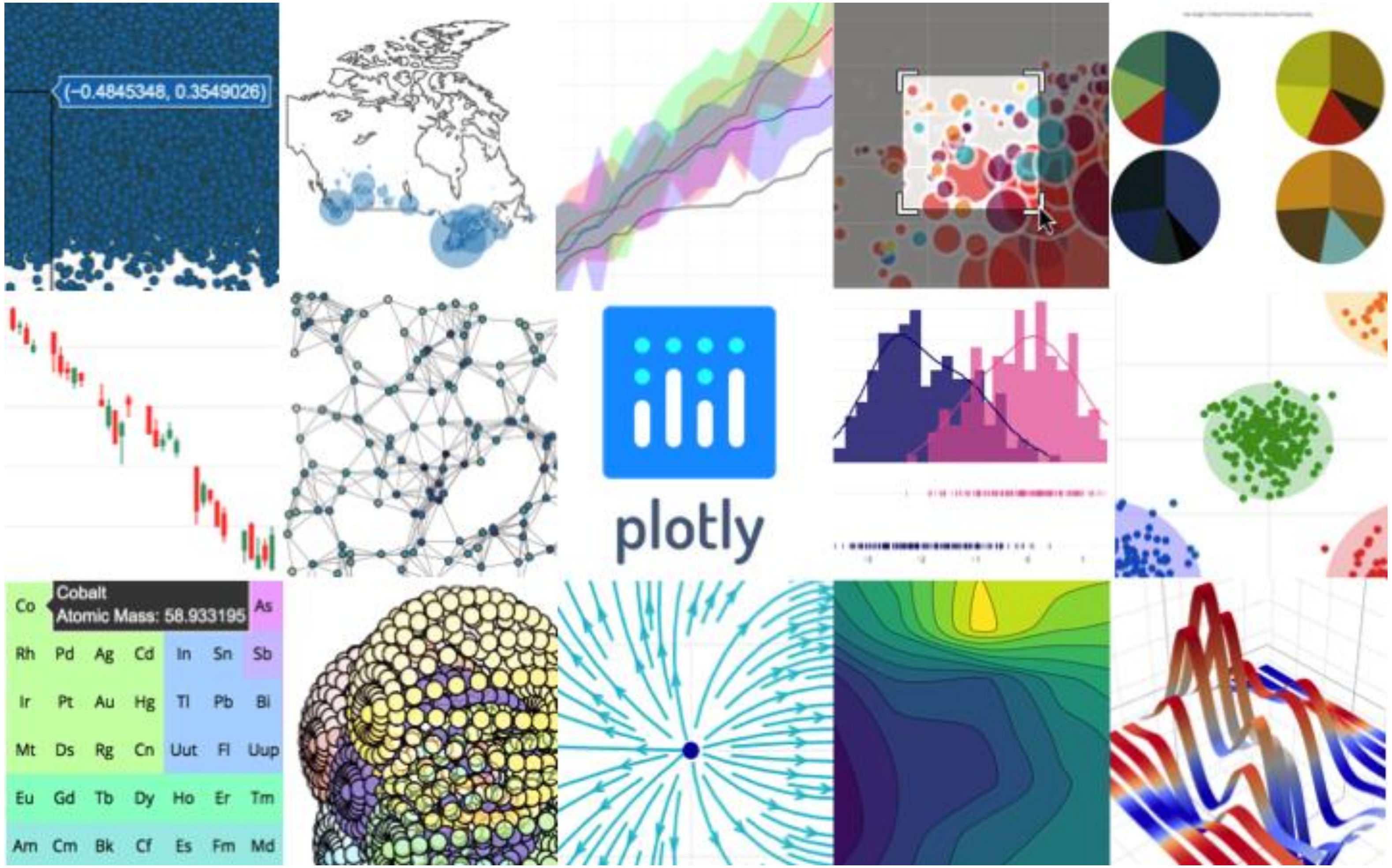
Interactive charting using FSharp.Plotly

CSB
COMPUTATIONAL SYSTEMS BIOLOGY

Kevin Schneider
Computational Systems Biology
Kaiserslautern University of Technology

Plotly.js

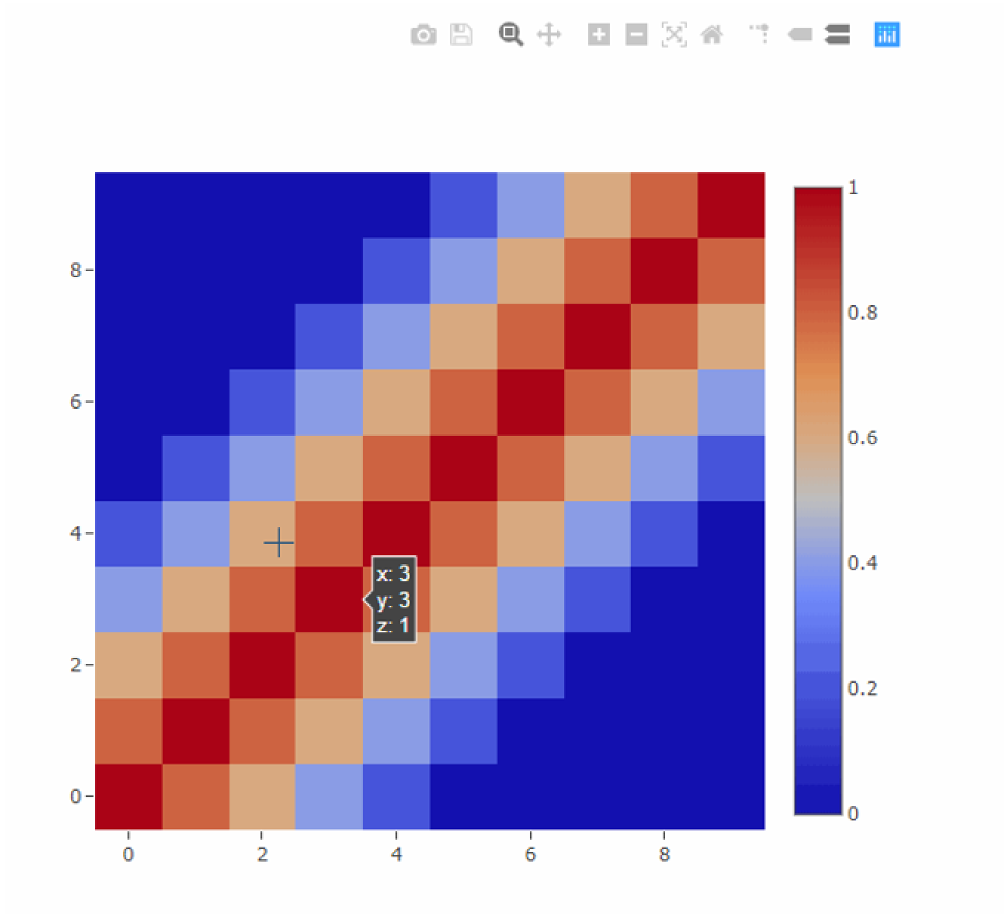
- Open source charting library
- Supports scientific charts, 3D graphs, statistical charts, etc.
- All charts are interactive



Computational Systems Biology

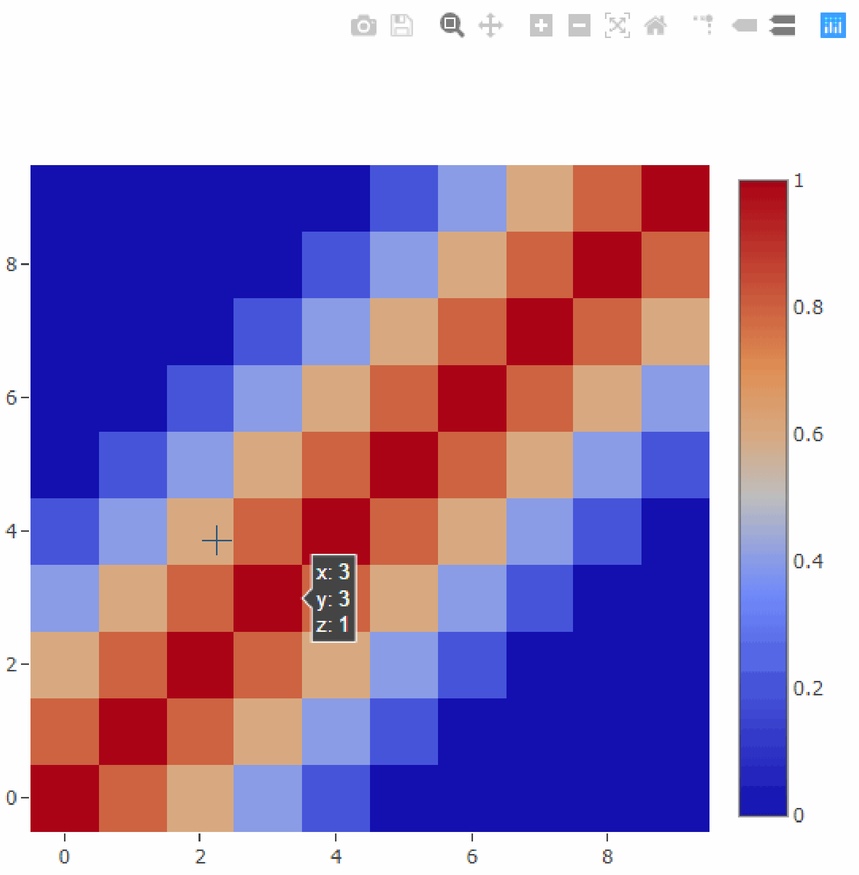
Plotly.js

- Plotly charts are interactive:



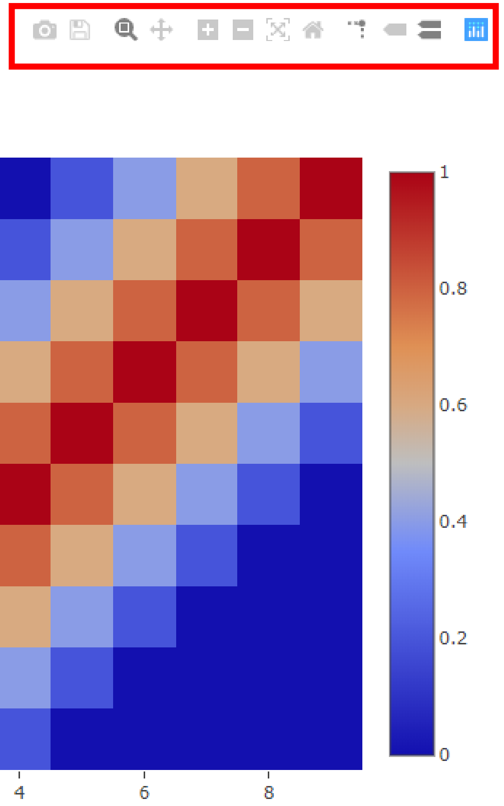
Plotly.js

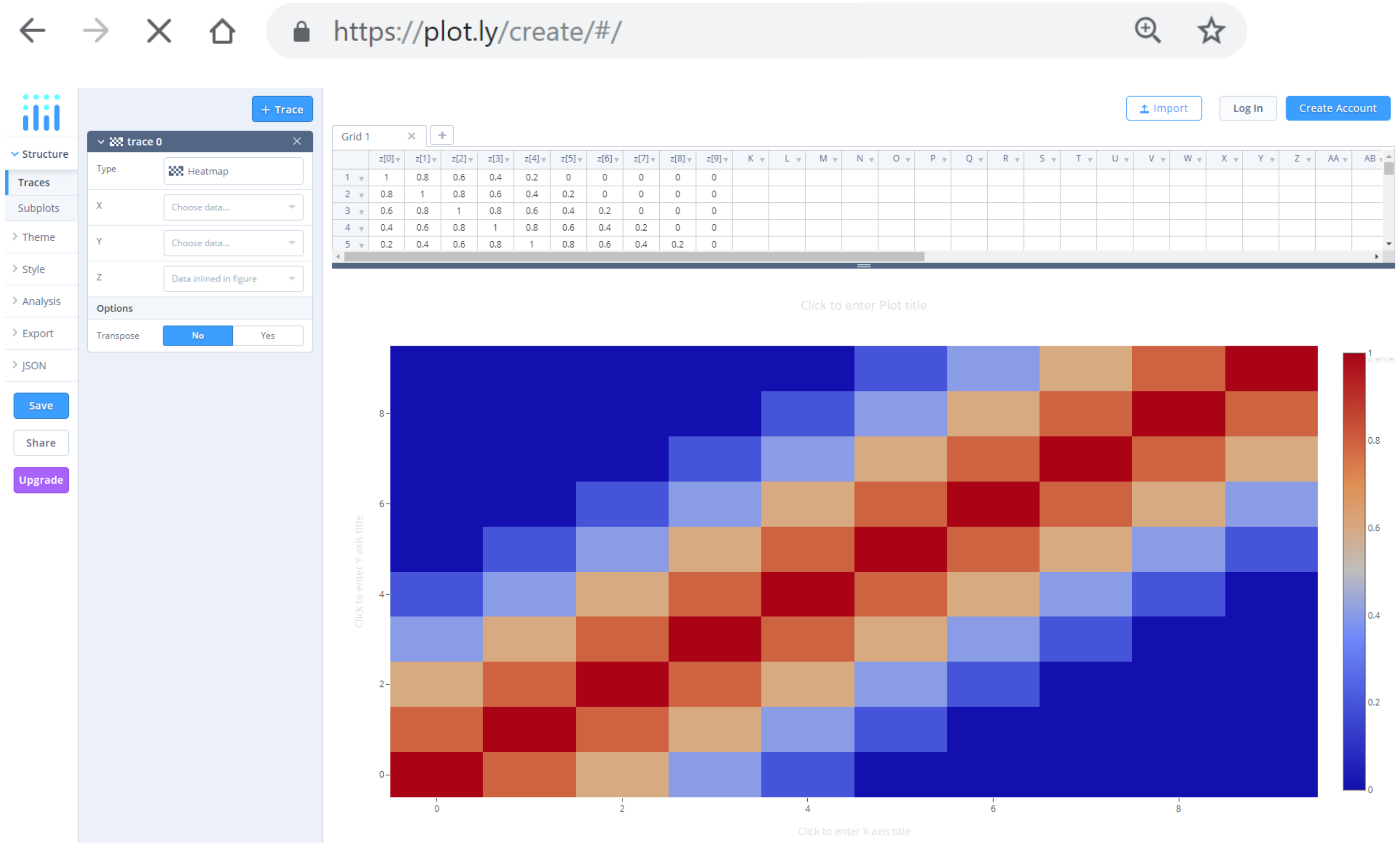
- Plotly charts are interactive:



Plotly.js

- Plotly.js is a service





trace 0

Type: Heatmap

X: Choose data...

Y: Choose data...

Z: Data inlined in figure

Options

Transpose: [✓ No] Yes

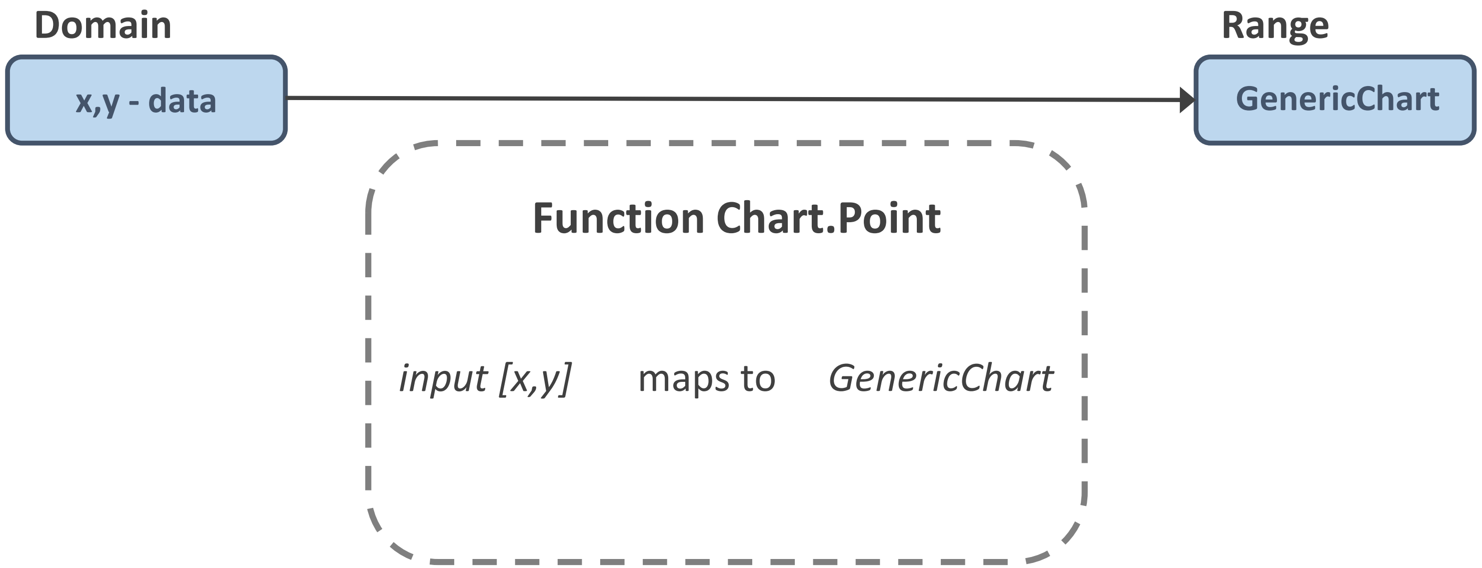
	z[0]	z[1]	z[2]	z[3]	z[4]	z[5]	z[6]	z[7]	z[8]	z[9]
1	1	0.8	0.6	0.4	0.2	0	0	0	0	0
2	0.8	1	0.8	0.6	0.4	0.2	0	0	0	0
3	0.6	0.8	1	0.8	0.6	0.4	0.2	0	0	0
4	0.4	0.6	0.8	1	0.8	0.6	0.4	0.2	0	0
5	0.2	0.4	0.6	0.8	1	0.8	0.6	0.4	0.2	0

FSharp.Plotly

- Open source F# wrapper for plotly.js
- Designed to work with many programming styles (object oriented, functional, mixtures,…)
- Documentation and tutorials can be found at:

http://muehlhaus.github.io/FSharp.Plotly/

Chart functions



Domain

x,y - data

Range

GenericChart

Function Chart.Point

input [x,y] maps to *GenericChart*

Chart functions

Domain

x,y - data

Range

GenericChart

```
[1.;2. ;3.;4. ;5.;6. ;7. ;8. ;9. ;10.; ]

[5.;2.5;5.;7.5;5.;2.5;7.5;4.5;5.5;5. ; ]
```

Function Chart.Point

input [x,y] maps to *GenericChart*

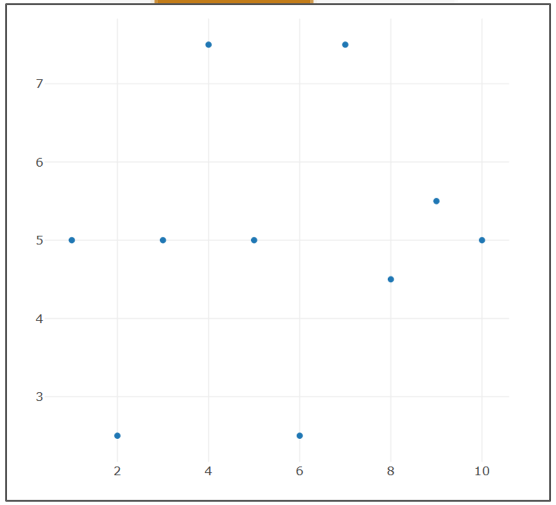
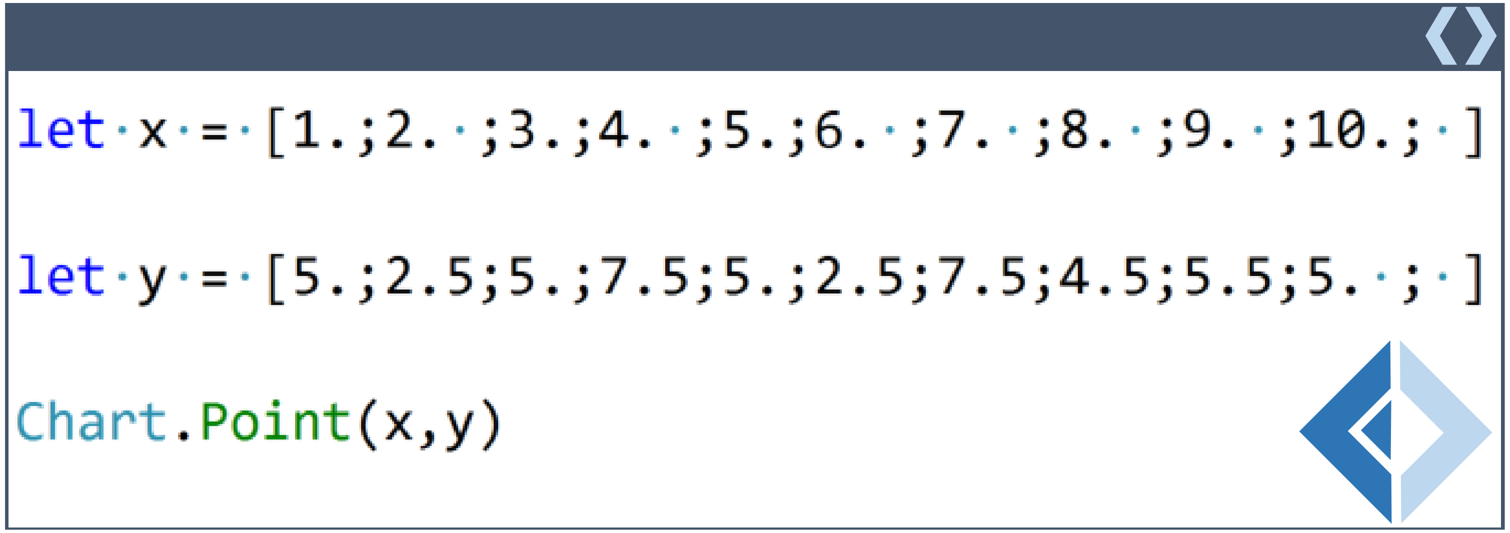
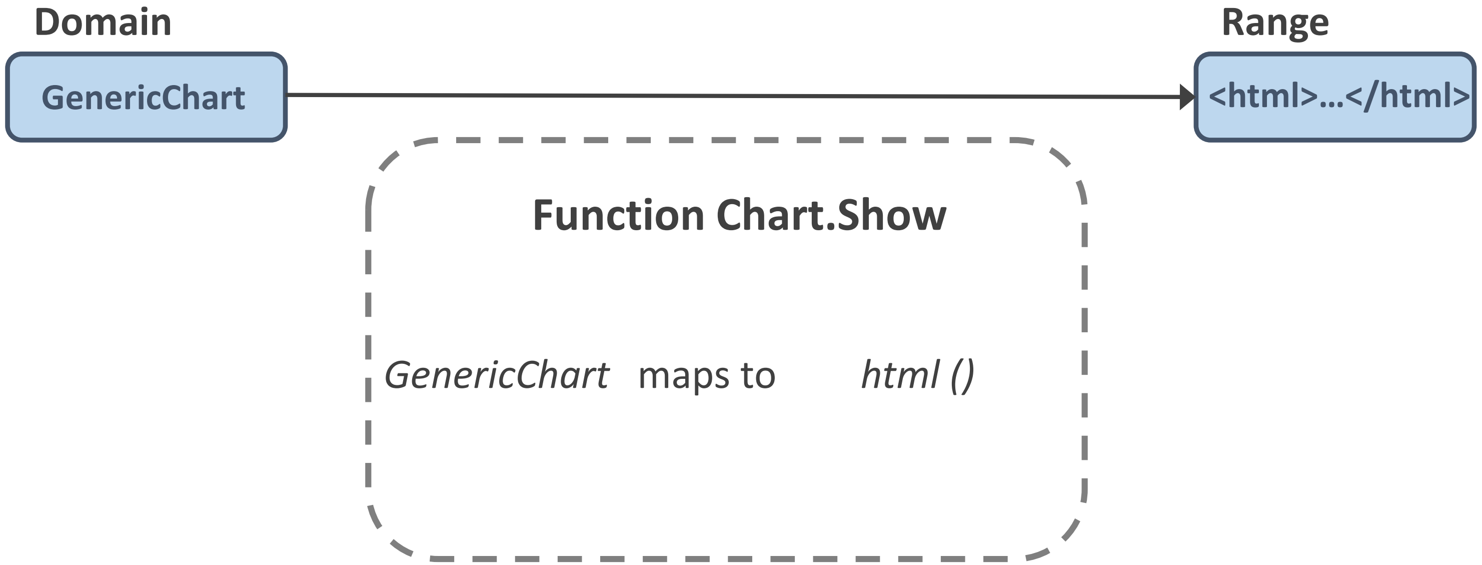


Chart functions - Example code



```
let x = [1.;2. ;3.;4. ;5.;6. ;7. ;8. ;9. ;10.; ]

let y = [5.;2.5;5.;7.5;5.;2.5;7.5;4.5;5.5;5. ; ]

Chart.Point(x,y)
```

Rendering your charts



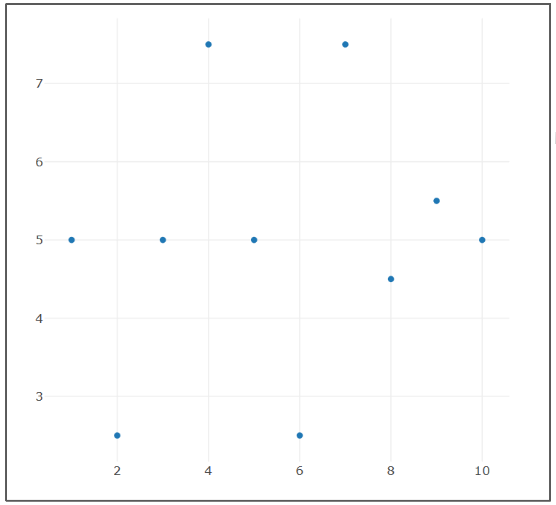
Rendering your charts

Domain

GenericChart

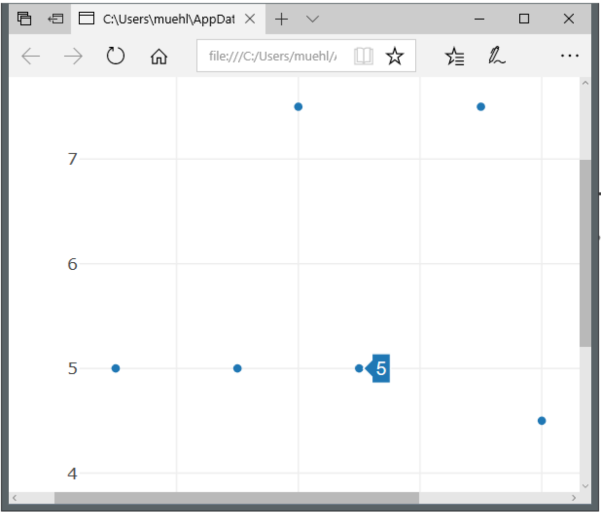
Range

<html>…</html>

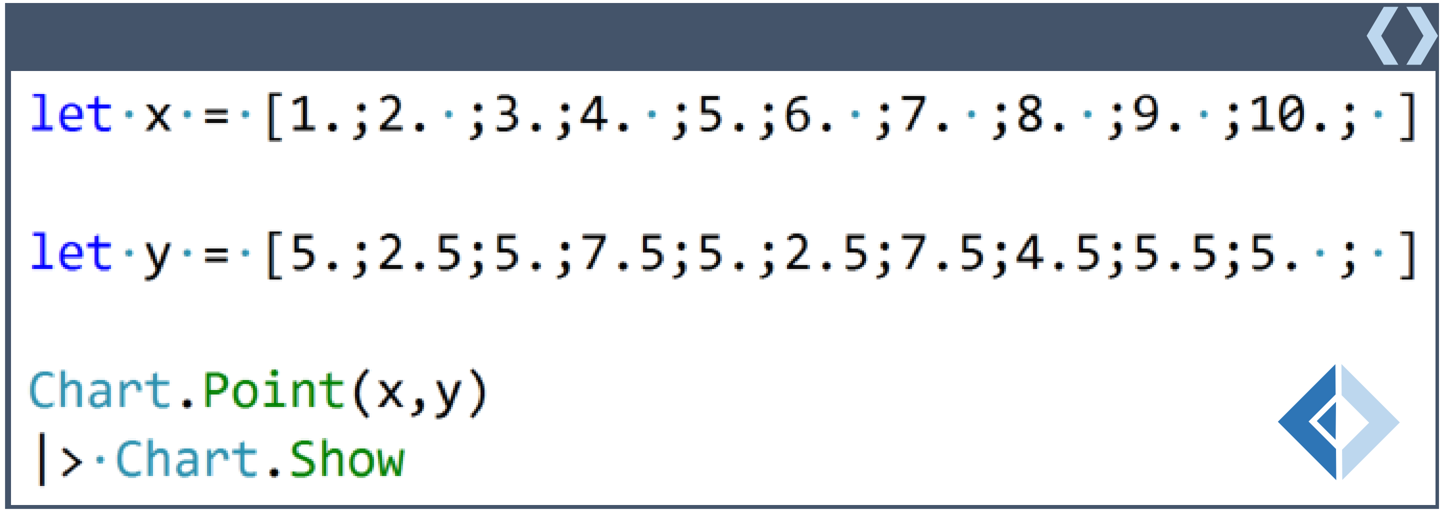


Function Chart.Show

GenericChart maps to *html ()*



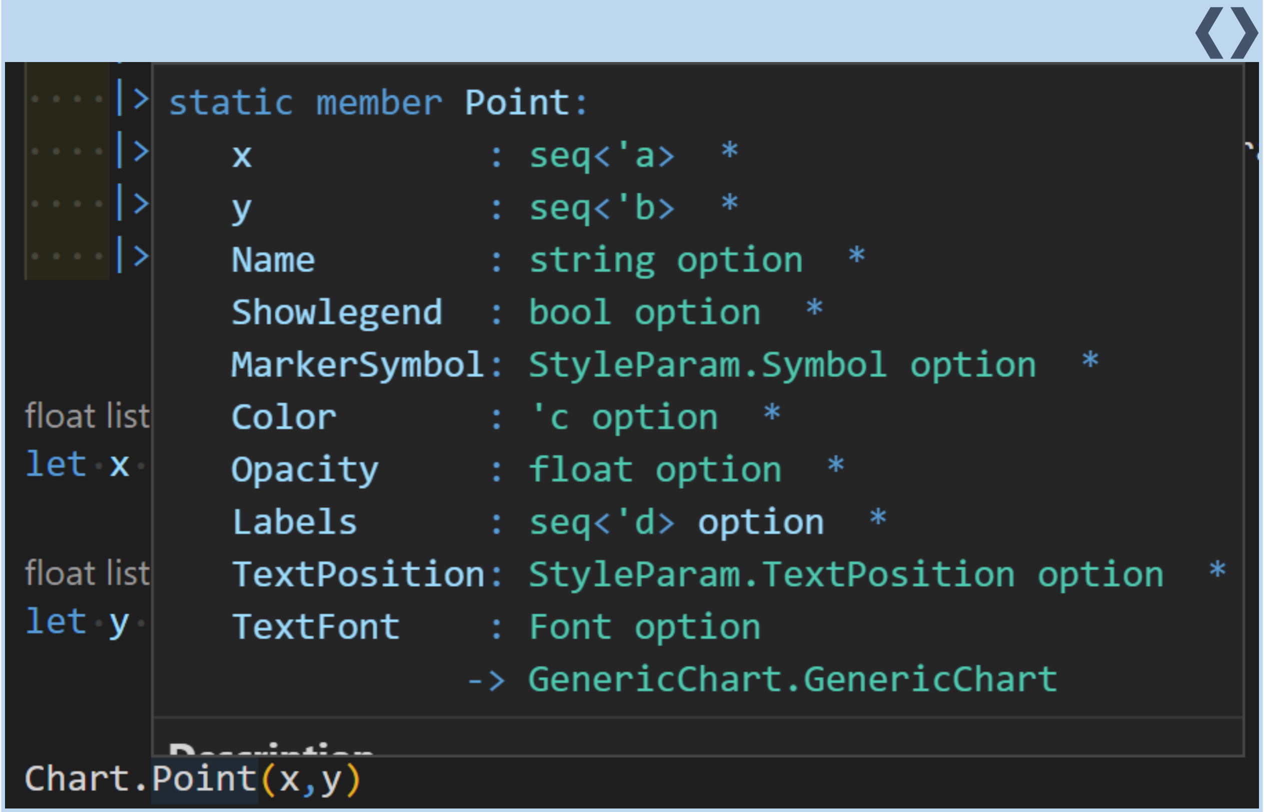
Rendering your charts - Example code



```
let x = [1.;2. ;3.;4. ;5.;6. ;7. ;8. ;9. ;10.; ]

let y = [5.;2.5;5.;7.5;5.;2.5;7.5;4.5;5.5;5. ; ]

Chart.Point(x,y)
|> Chart.Show
```

Style your charts

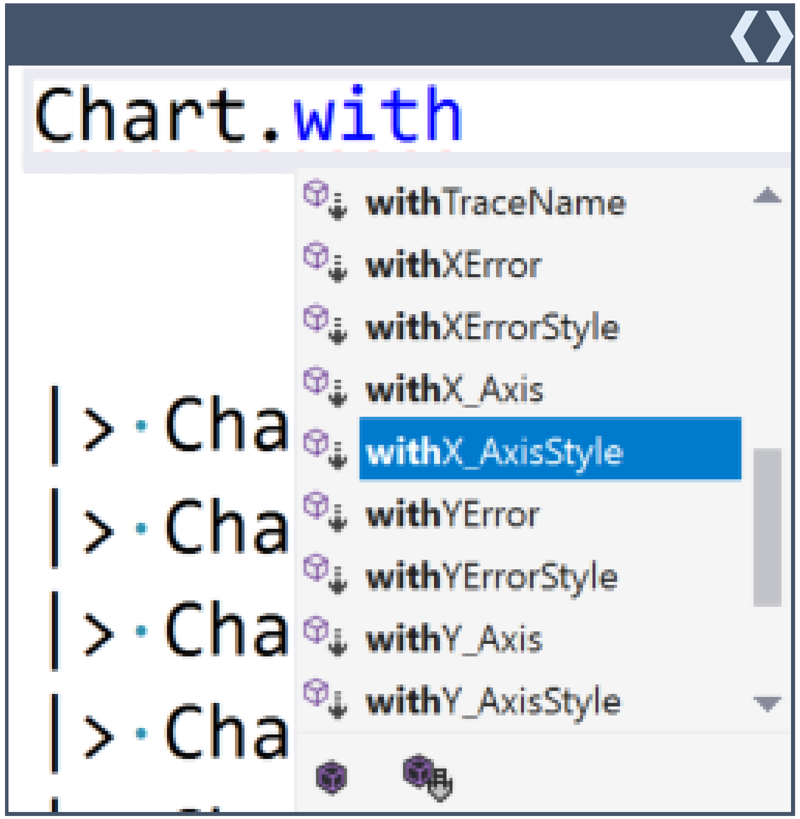
Additional parameters for chart functions:



```
static member Point:
    x           : seq<'a>  *
    y           : seq<'b>  *
    Name        : string option  *
    Showlegend  : bool option  *
    MarkerSymbol: StyleParam.Symbol option  *
    Color       : 'c option  *
    Opacity     : float option  *
    Labels      : seq<'d> option  *
    TextPosition: StyleParam.TextPosition option  *
    TextFont    : Font option
               -> GenericChart.GenericChart
```

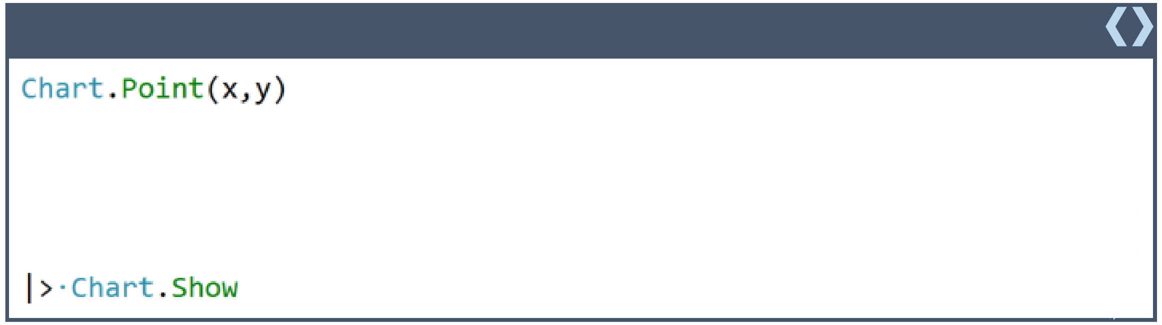
```
float list
let x

float list
let y

Chart.Point(x,y)
```

Chart extension functions:

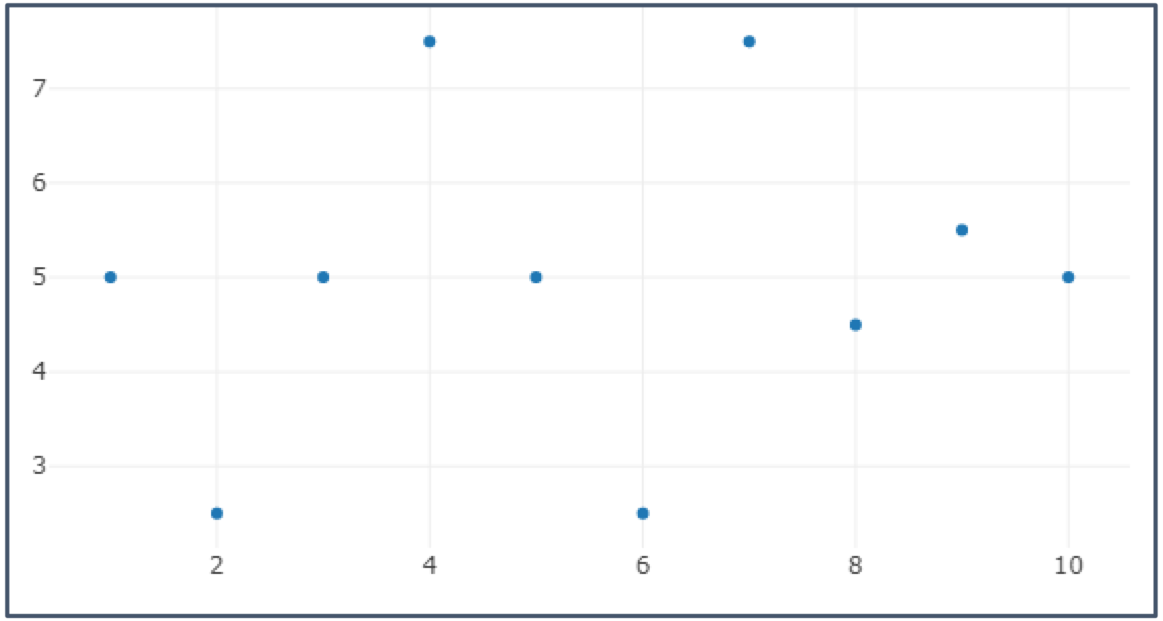


```
Chart.with
```

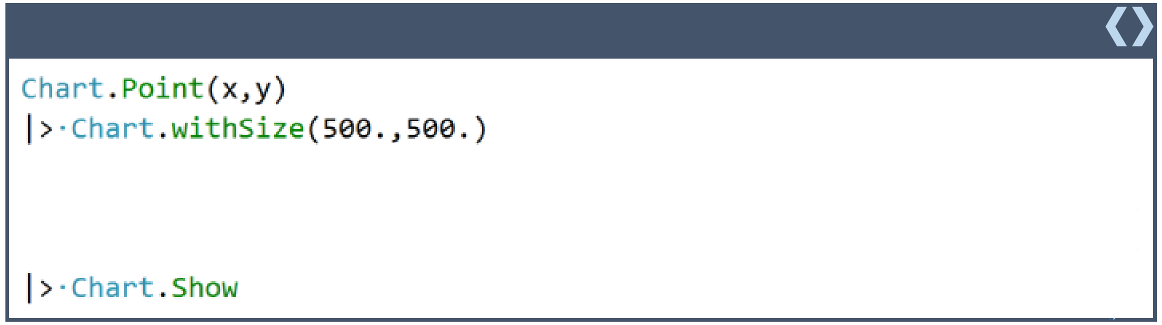
Style your charts - Example code



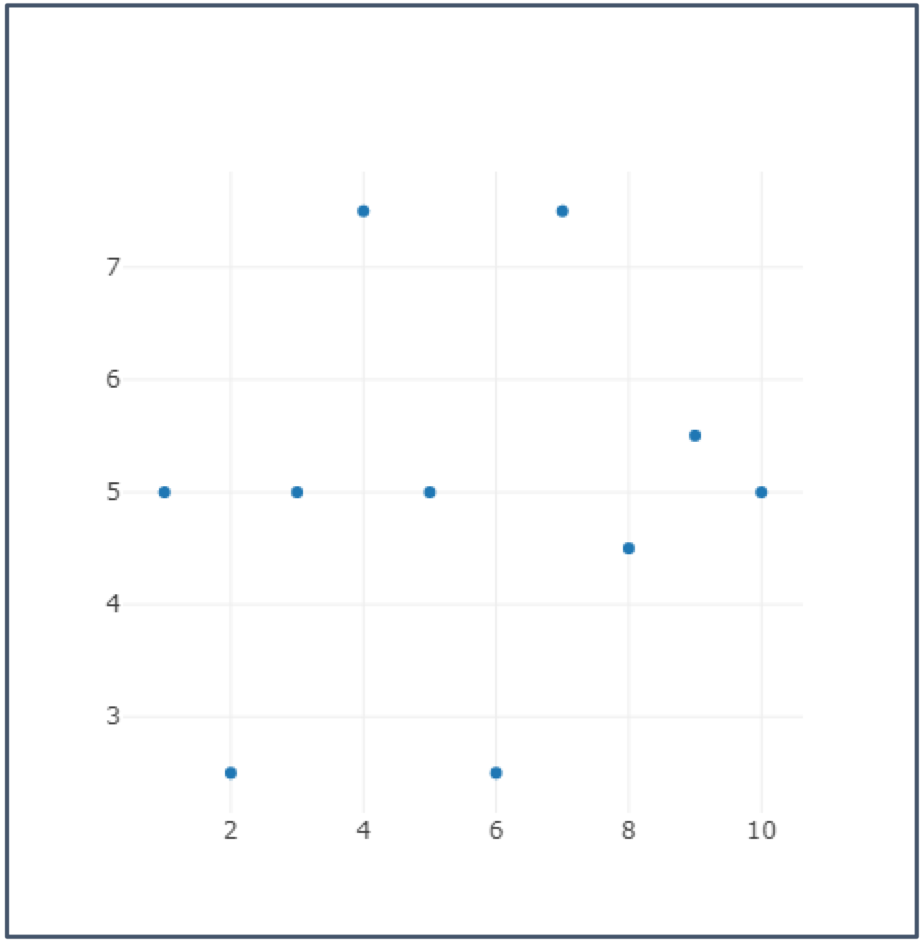
```
Chart.Point(x,y)



|> Chart.Show
```



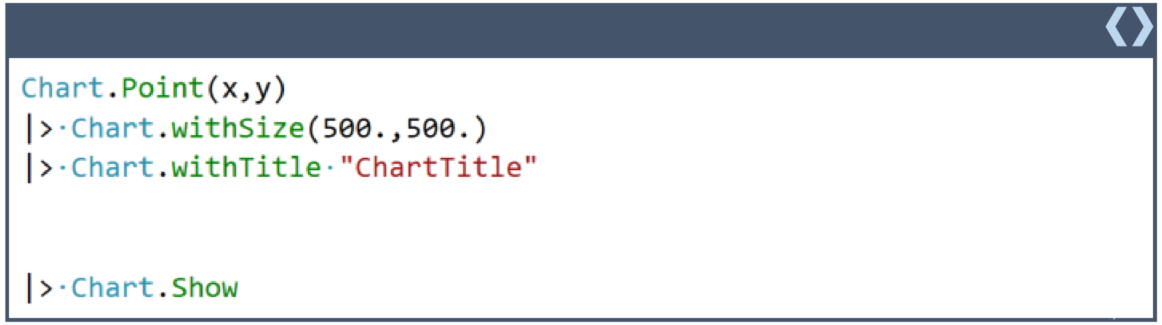
Style your charts - Example code



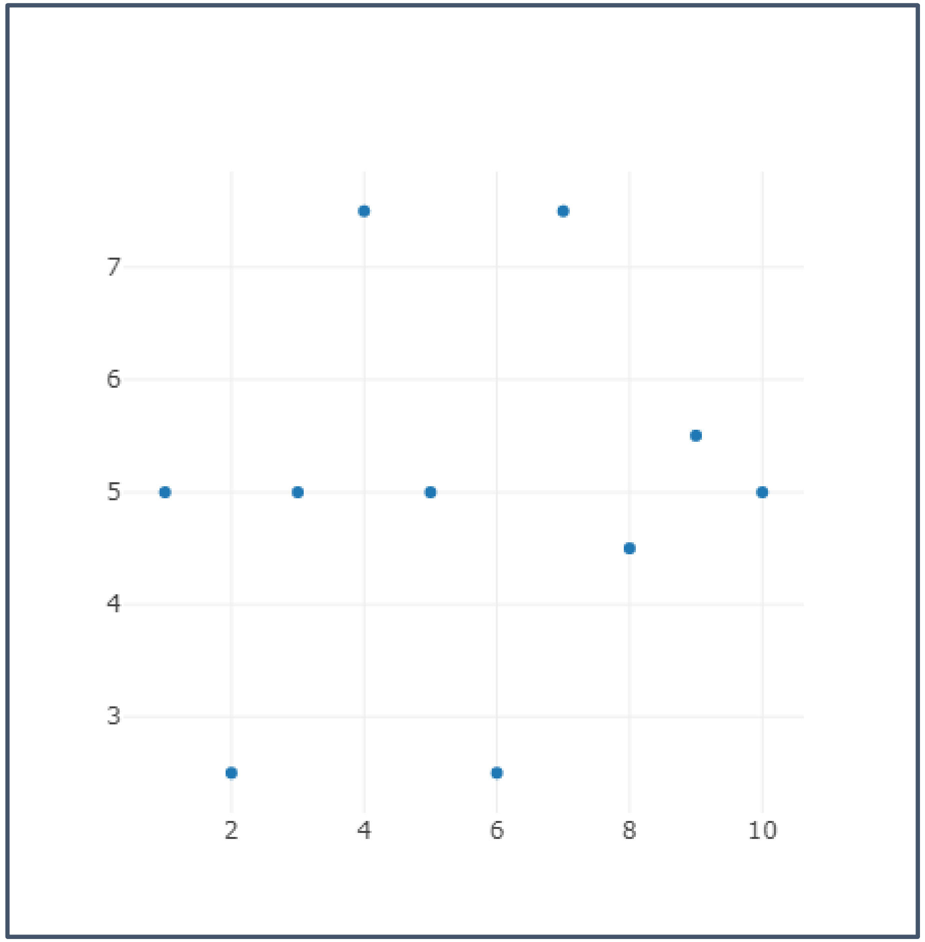
```
Chart.Point(x,y)
|>·Chart.withSize(500.,500.)



|>·Chart.Show
```



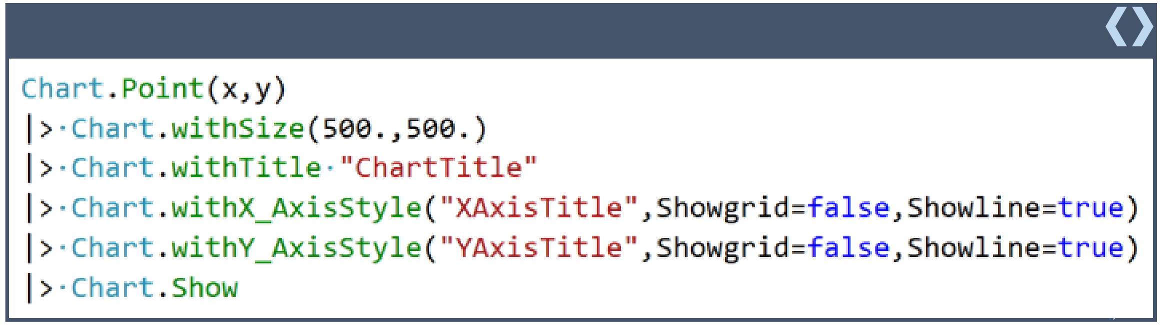
Style your charts - Example code



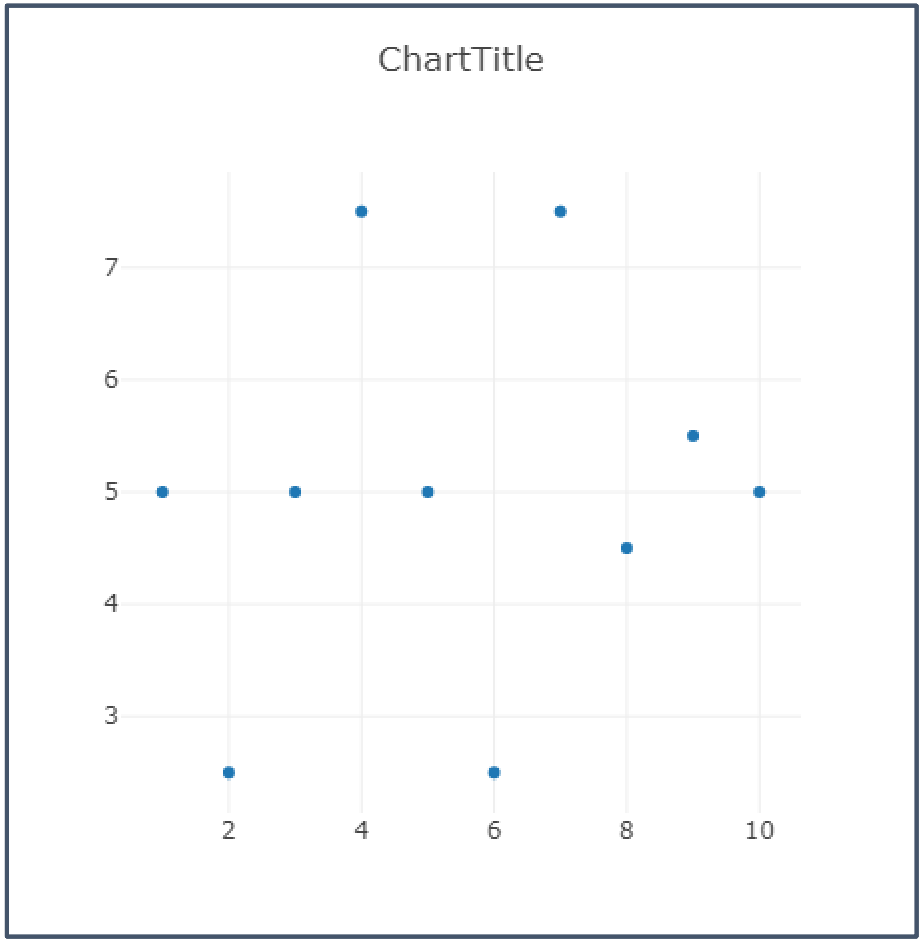
```
Chart.Point(x,y)
|> Chart.withSize(500.,500.)
|> Chart.withTitle "ChartTitle"


|> Chart.Show
```



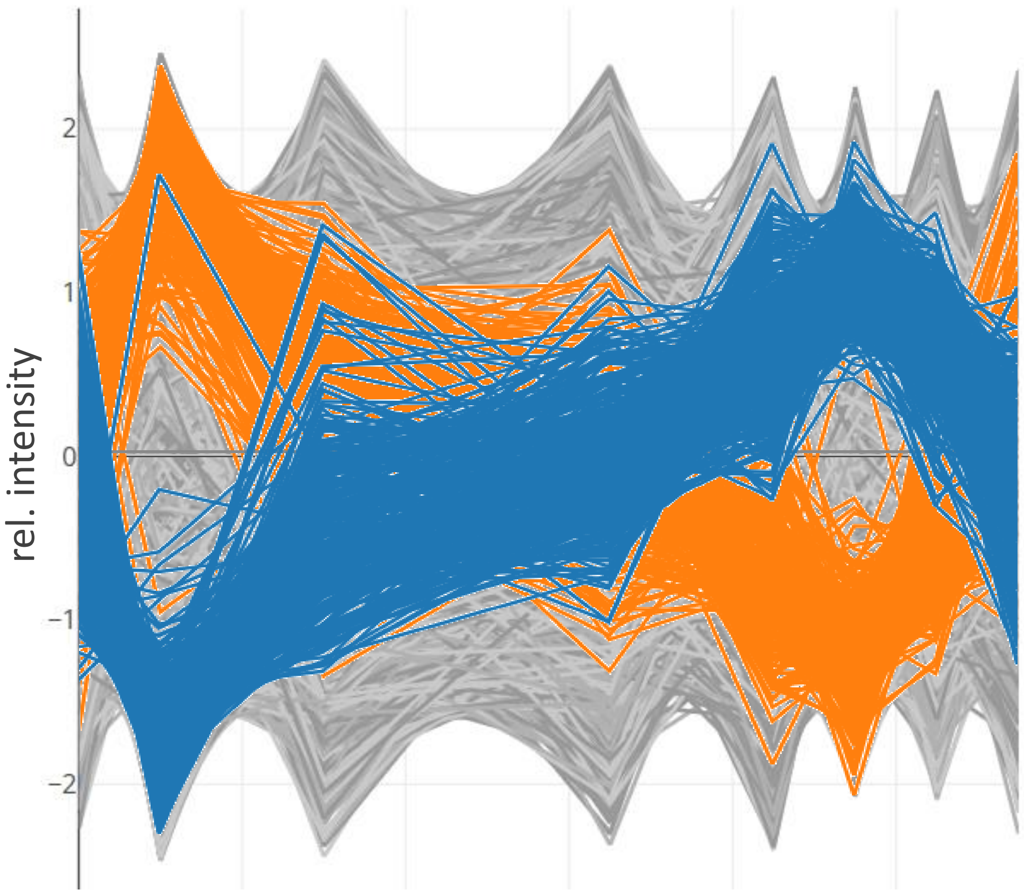
Style your charts - Example code

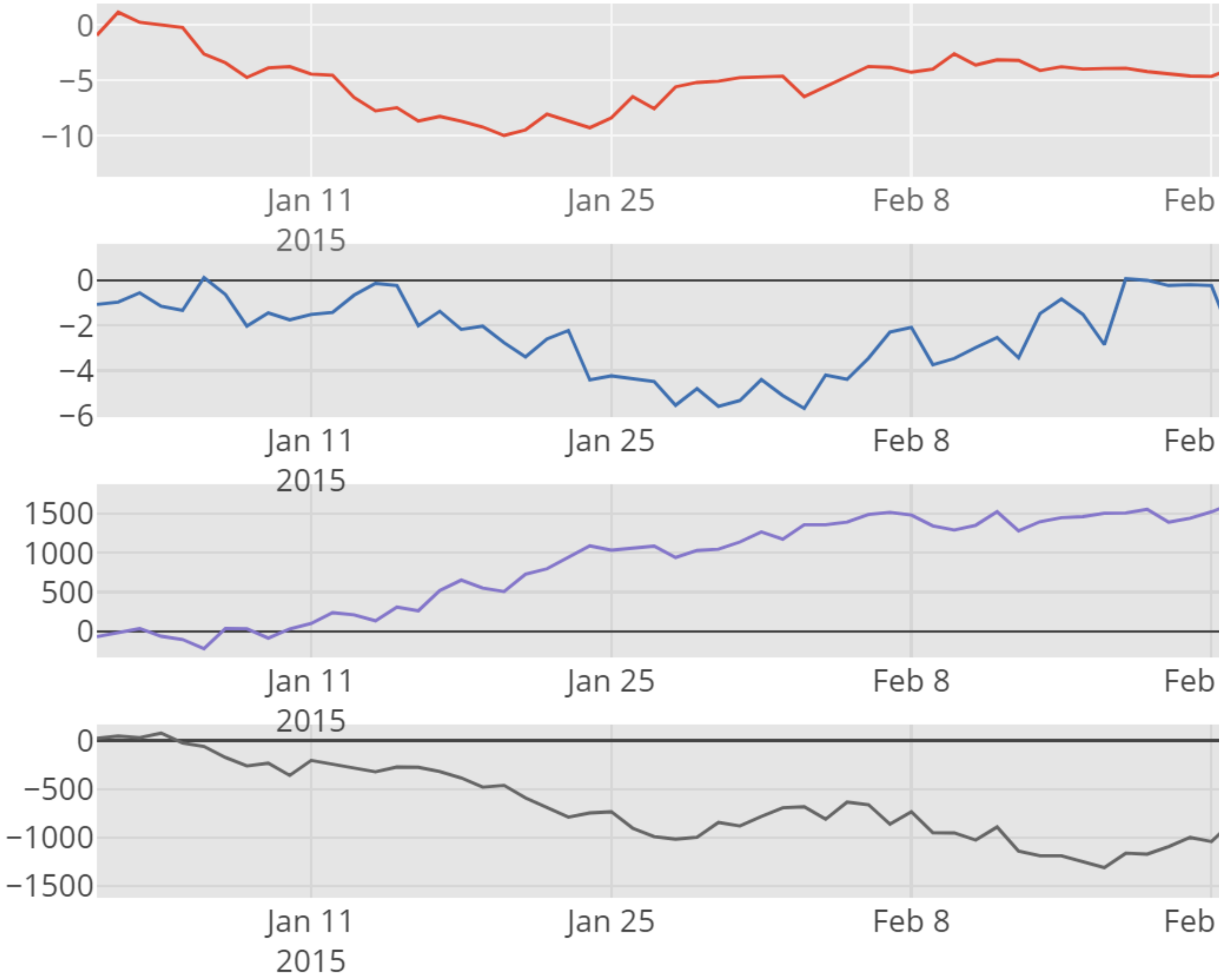


```
Chart.Point(x,y)
|>·Chart.withSize(500.,500.)
|>·Chart.withTitle·"ChartTitle"
|>·Chart.withX_AxisStyle("XAxisTitle",Showgrid=false,Showline=true)
|>·Chart.withY_AxisStyle("YAxisTitle",Showgrid=false,Showline=true)
|>·Chart.Show
```

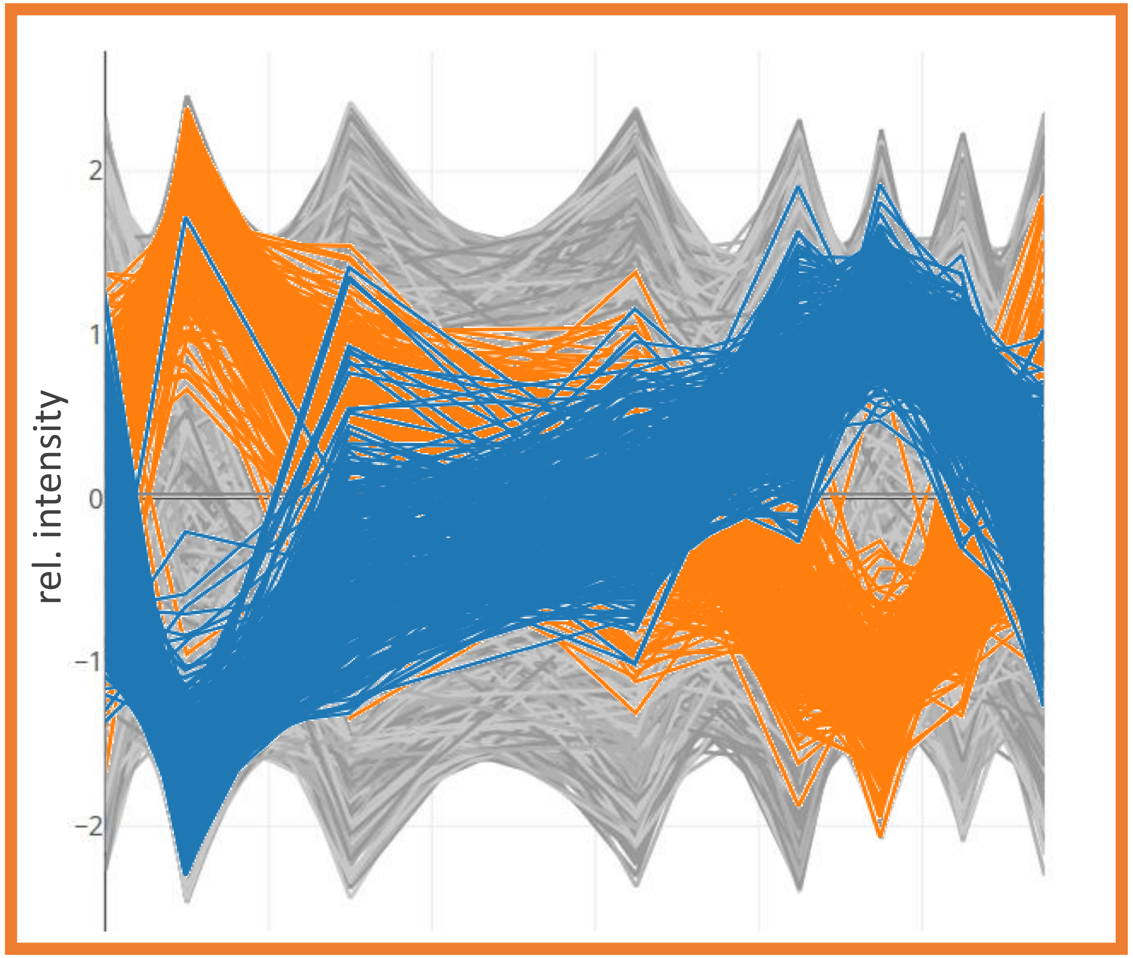


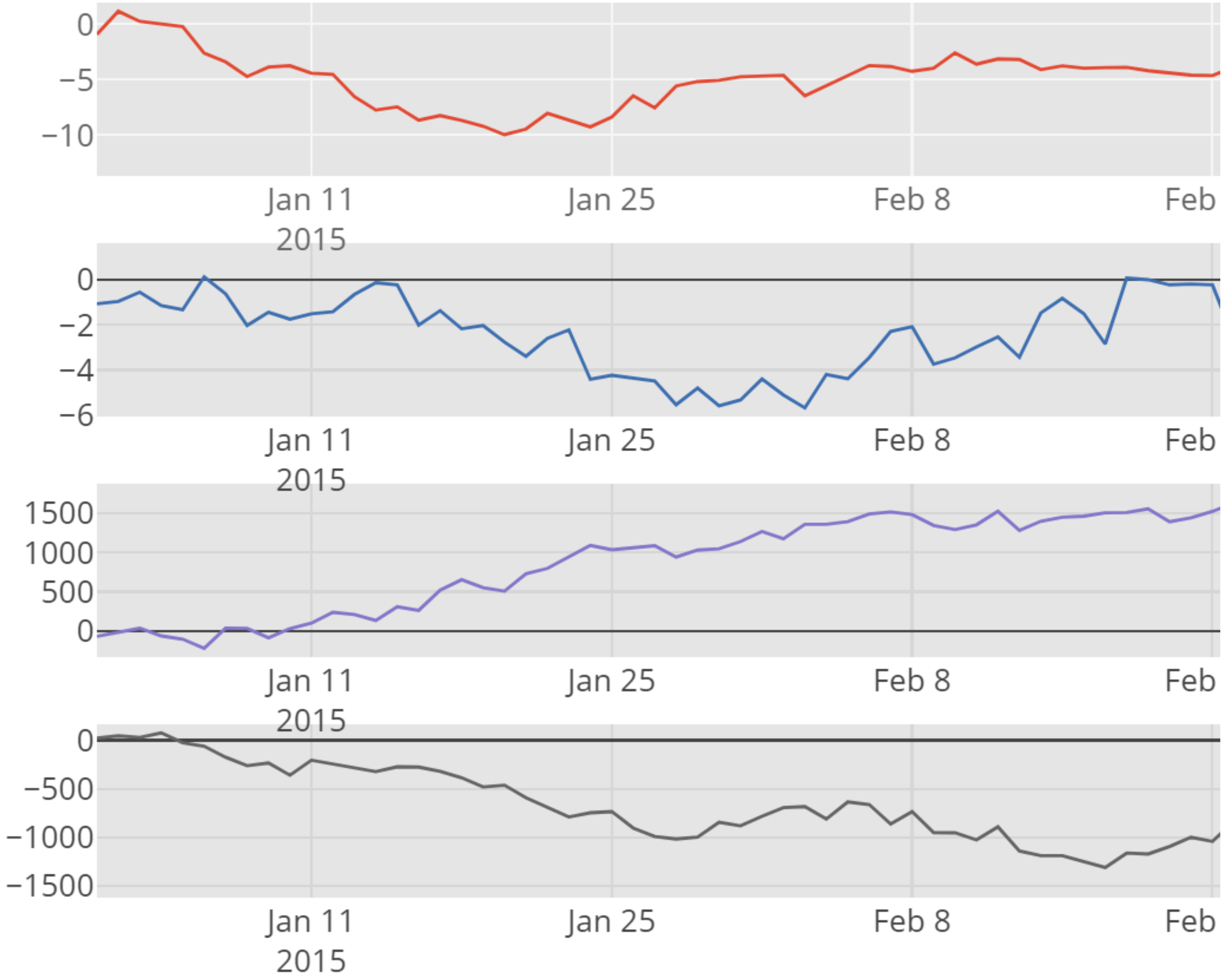
Multicharts





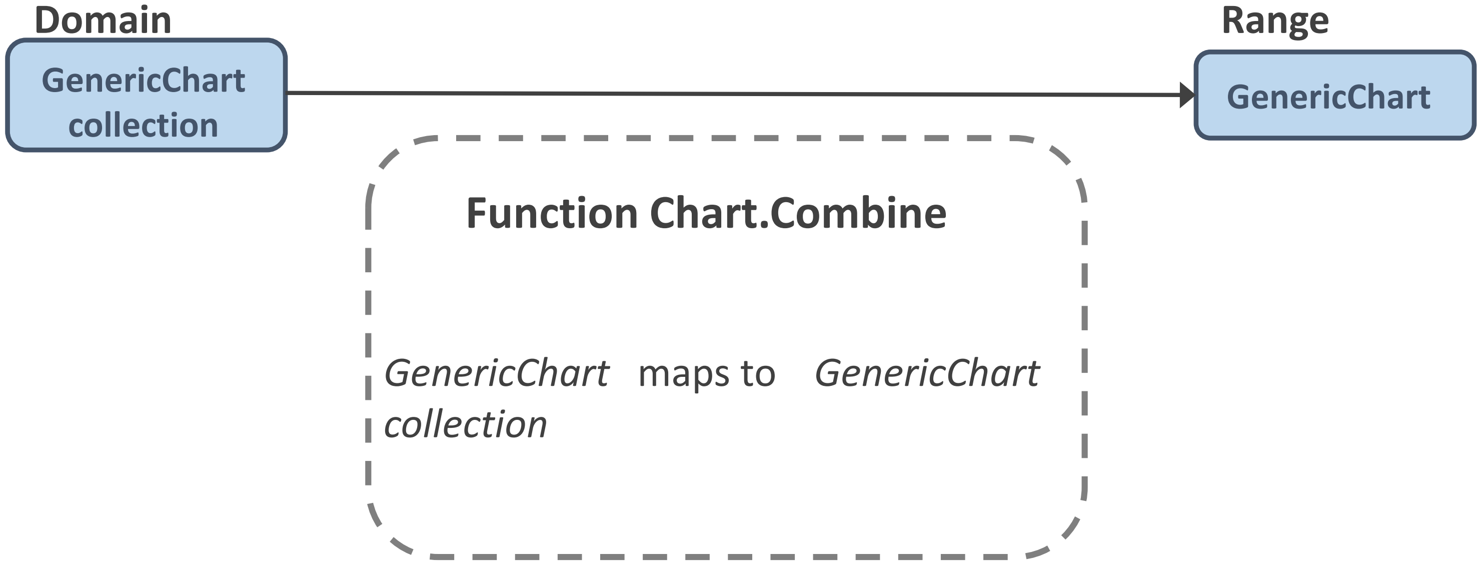
Multicharts – Combine Charts





Computational
Systems Biology

Combine Charts – Chart.Combine



GenericChart collection maps to *GenericChart*

Combine Charts – Chart.Combine

Domain

GenericChart collection

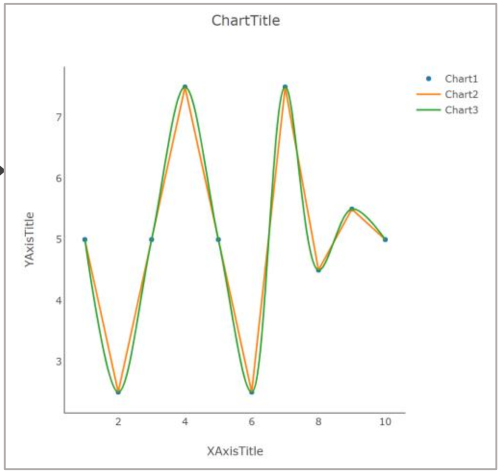
Range

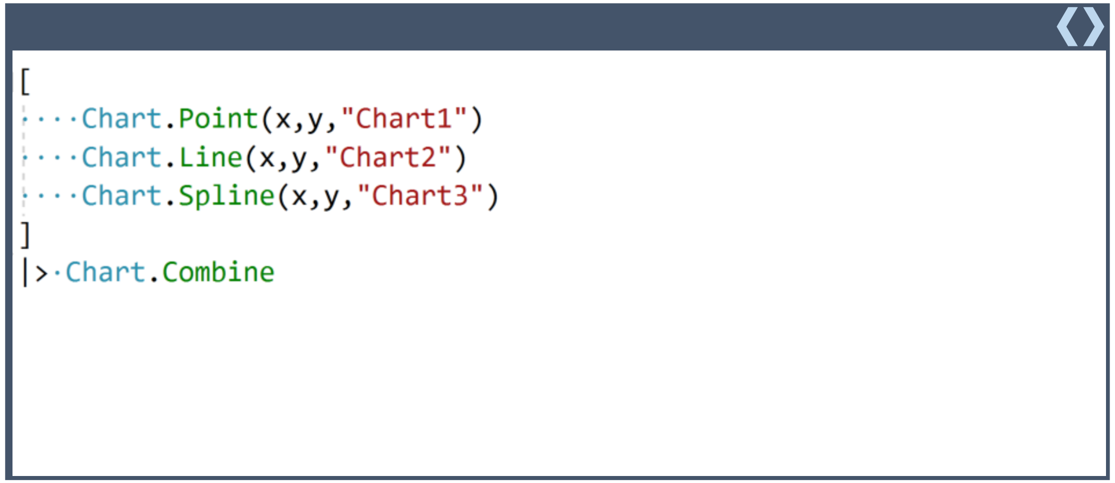
GenericChart

```
[
  Chart.Point(...)
  Chart.Line(...)
  Chart. ...
  ...
]
```

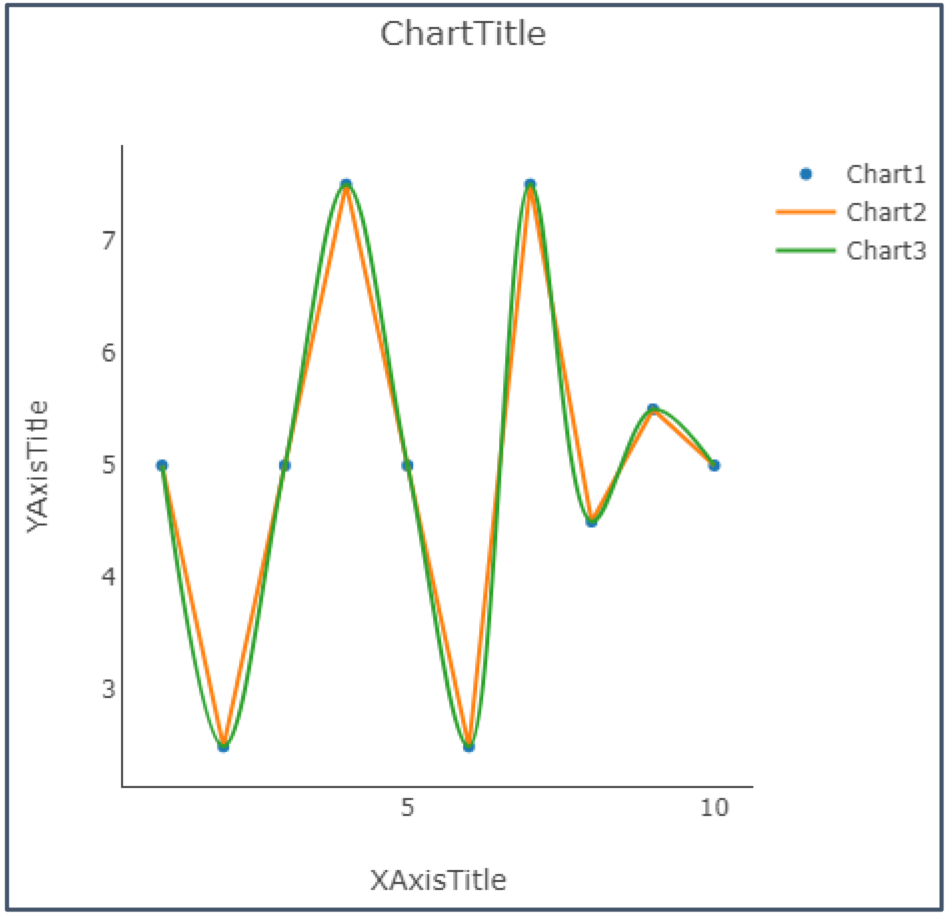
Function Chart.Combine

GenericChart collection maps to *GenericChart*

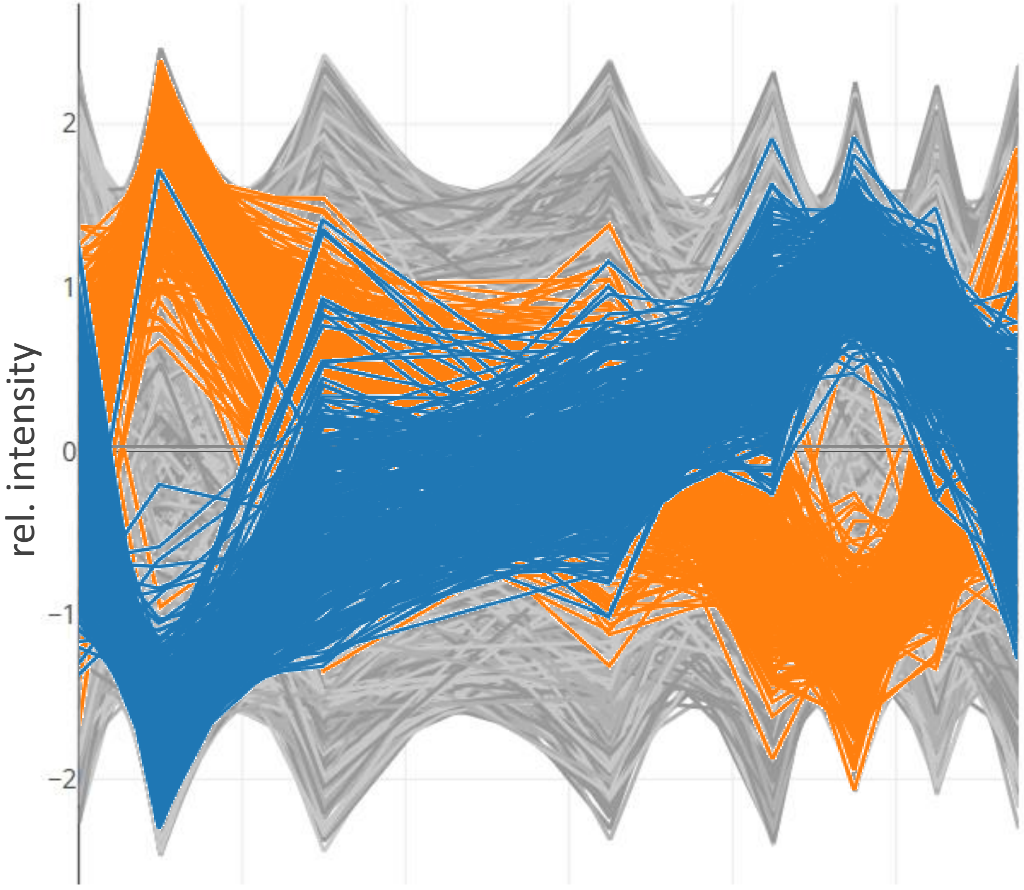


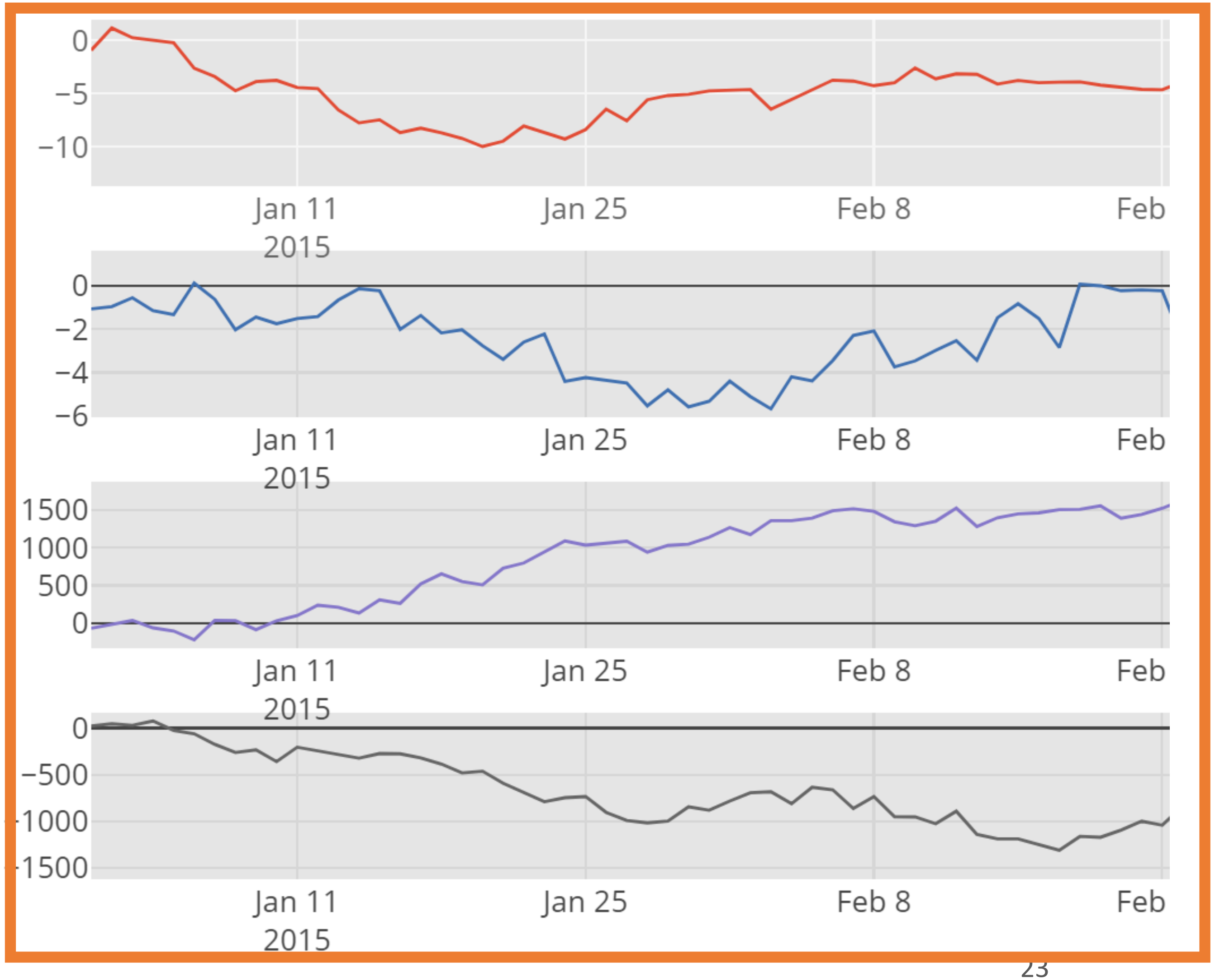


```
[
    Chart.Point(x,y,"Chart1")
    Chart.Line(x,y,"Chart2")
    Chart.Spline(x,y,"Chart3")
]
|> Chart.Combine
```

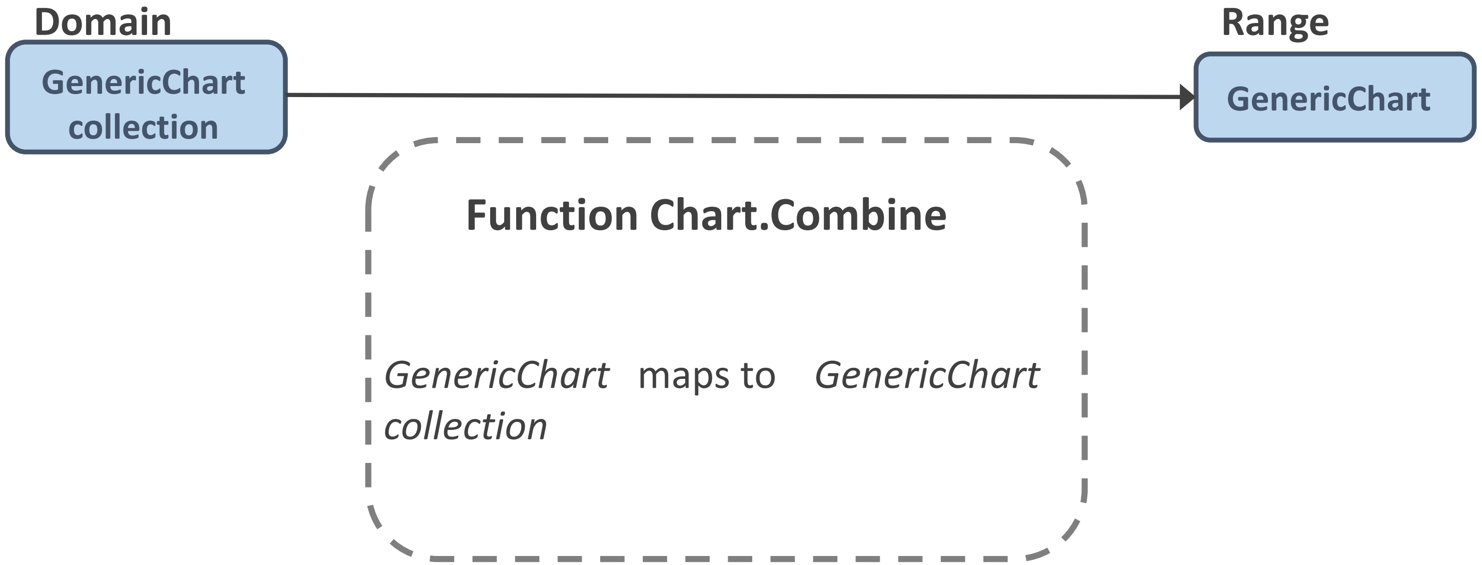


Multicharts – Stack Charts





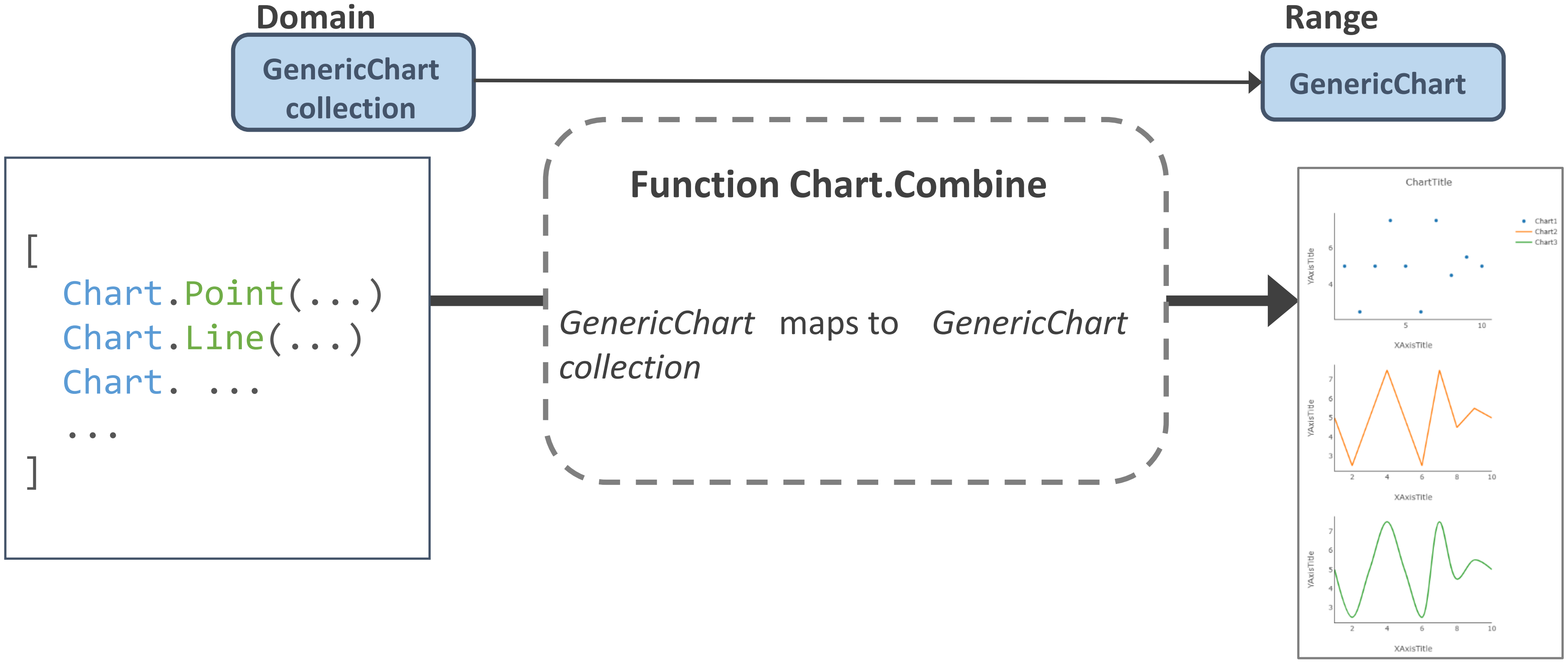
Combine Charts – Chart.Stack



Function Chart.Combine

GenericChart collection maps to *GenericChart*

Combine Charts – Chart.Stack



Domain

GenericChart collection

Range

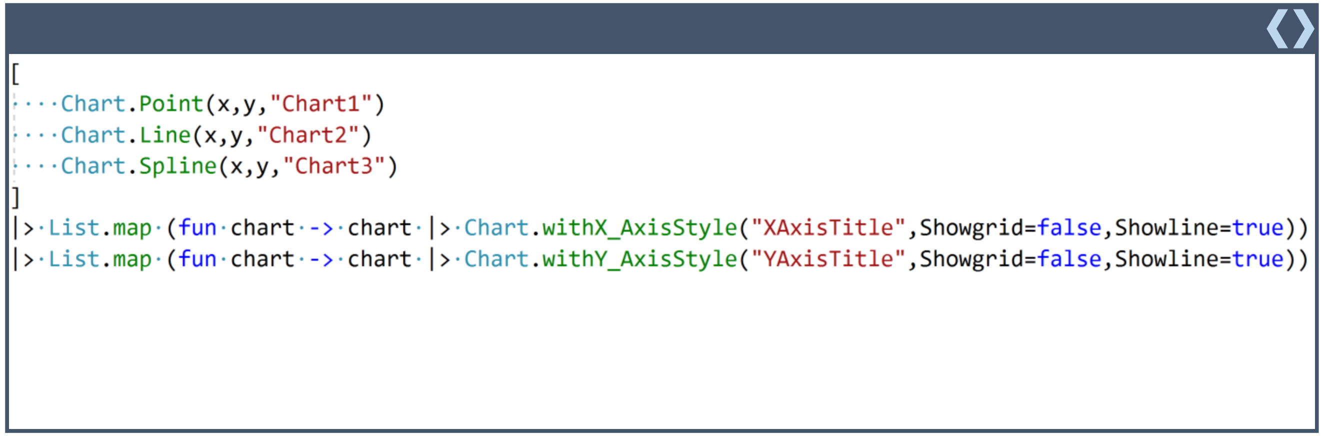
GenericChart

```
[
  Chart.Point(...)
  Chart.Line(...)
  Chart. ...
  ...
]
```

Function Chart.Combine

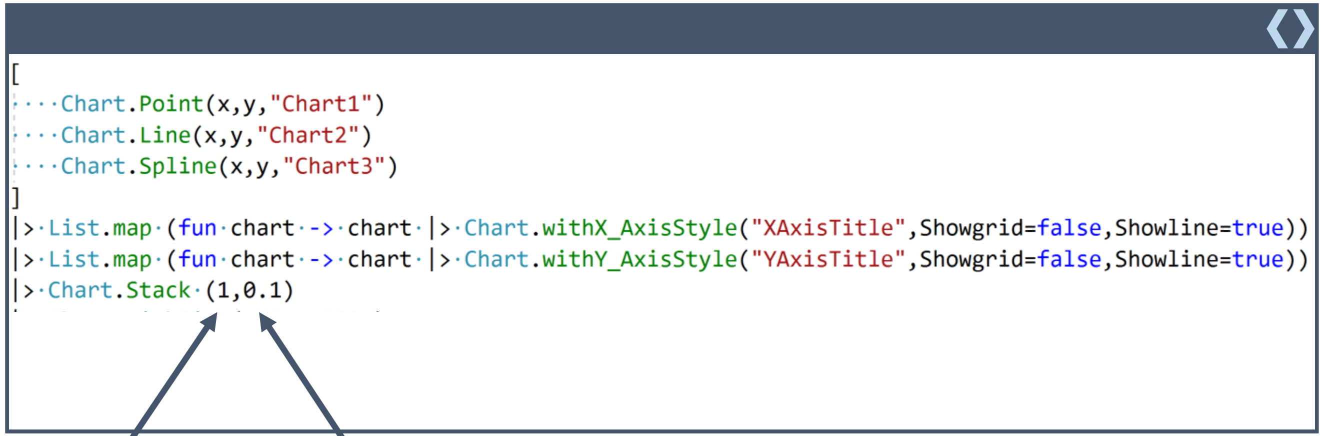
GenericChart collection maps to *GenericChart*

Combine Charts – Chart.Stack



```
[
    Chart.Point(x,y,"Chart1")
    Chart.Line(x,y,"Chart2")
    Chart.Spline(x,y,"Chart3")
]
|> List.map (fun chart -> chart |> Chart.withX_AxisStyle("XAxisTitle",Showgrid=false,Showline=true))
|> List.map (fun chart -> chart |> Chart.withY_AxisStyle("YAxisTitle",Showgrid=false,Showline=true))
```

Combine Charts – Chart.Stack

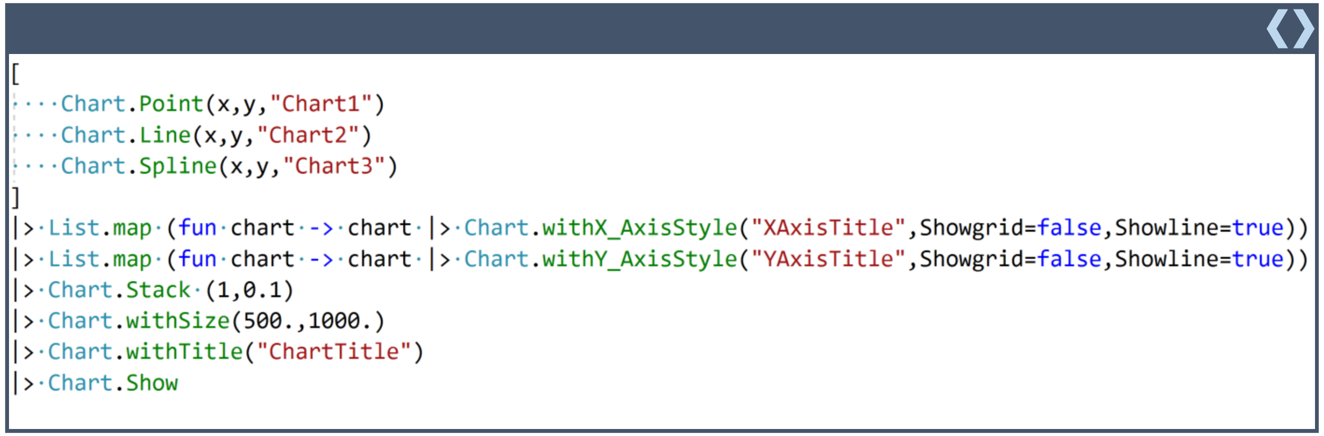


```
[
    Chart.Point(x,y,"Chart1")
    Chart.Line(x,y,"Chart2")
    Chart.Spline(x,y,"Chart3")
]
|> List.map (fun chart -> chart |> Chart.withX_AxisStyle("XAxisTitle",Showgrid=false,Showline=true))
|> List.map (fun chart -> chart |> Chart.withY_AxisStyle("YAxisTitle",Showgrid=false,Showline=true))
|> Chart.Stack (1,0.1)
```

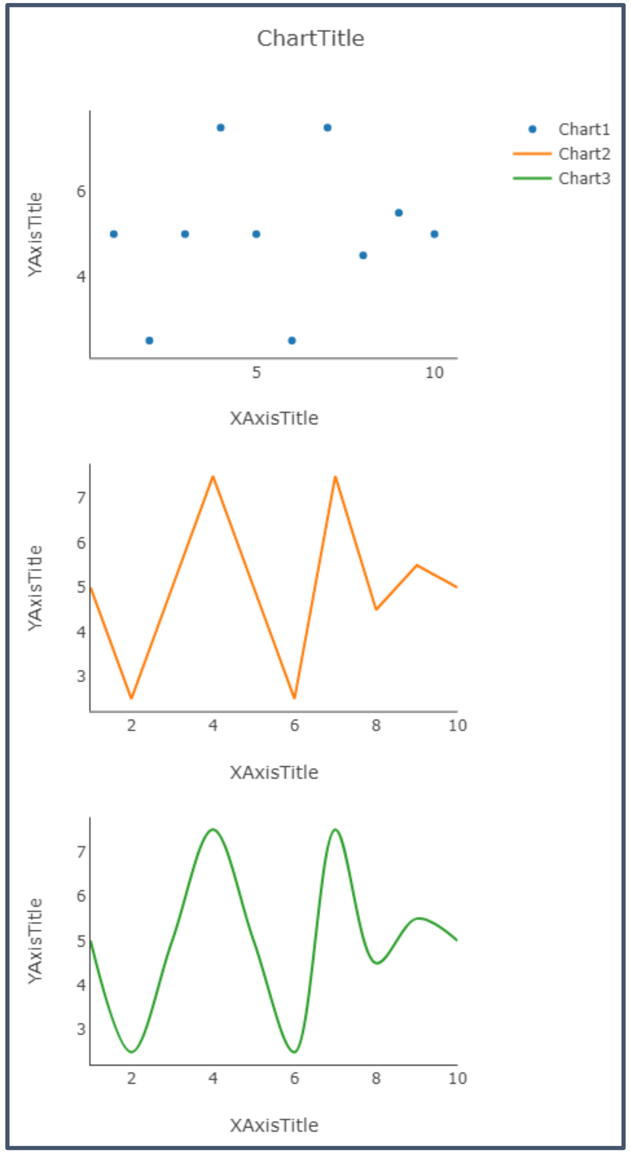
Amount of columns

Spacing between subplots

Combine Charts – Chart.Stack

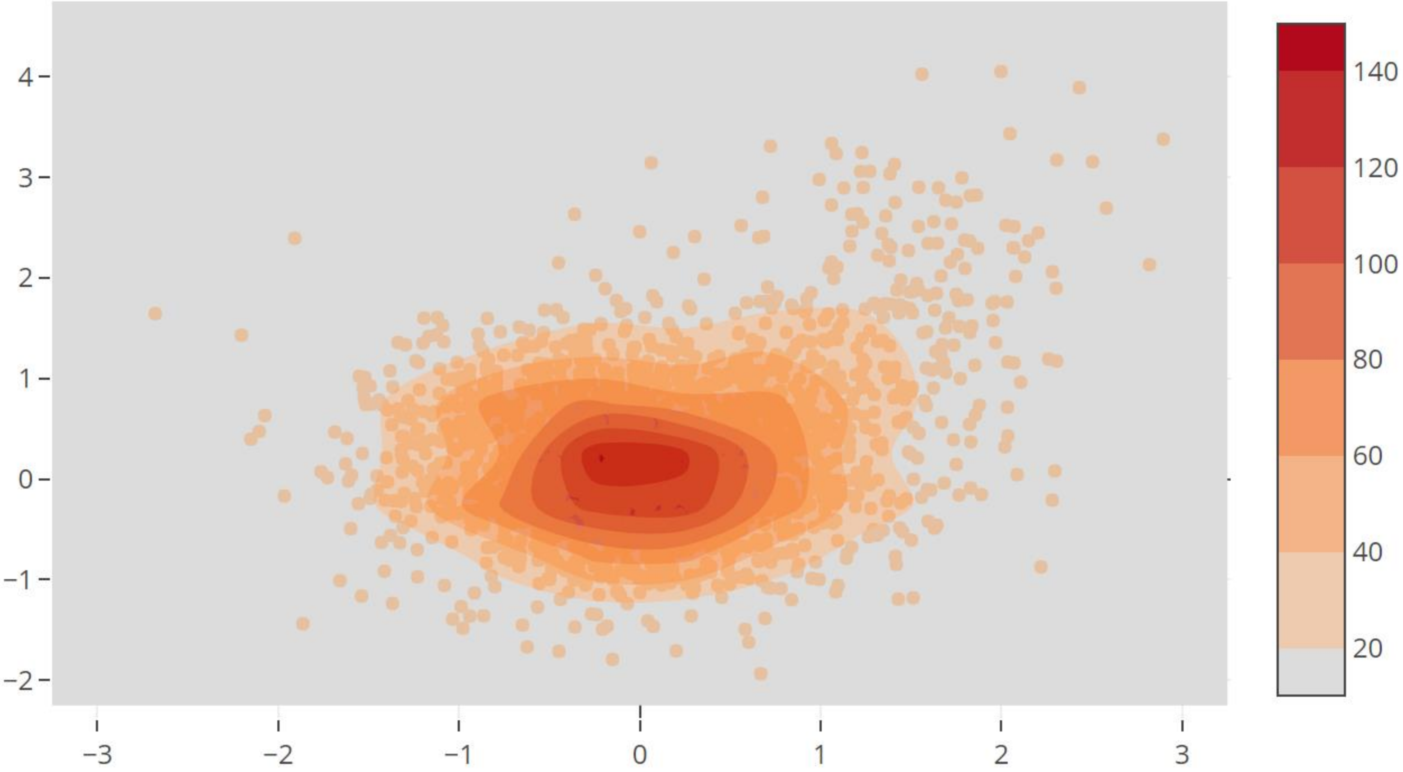


```
[
    Chart.Point(x,y,"Chart1")
    Chart.Line(x,y,"Chart2")
    Chart.Spline(x,y,"Chart3")
]
|> List.map (fun chart -> chart |> Chart.withX_AxisStyle("XAxisTitle",Showgrid=false,Showline=true))
|> List.map (fun chart -> chart |> Chart.withY_AxisStyle("YAxisTitle",Showgrid=false,Showline=true))
|> Chart.Stack (1,0.1)
|> Chart.withSize(500.,1000.)
|> Chart.withTitle("ChartTitle")
|> Chart.Show
```

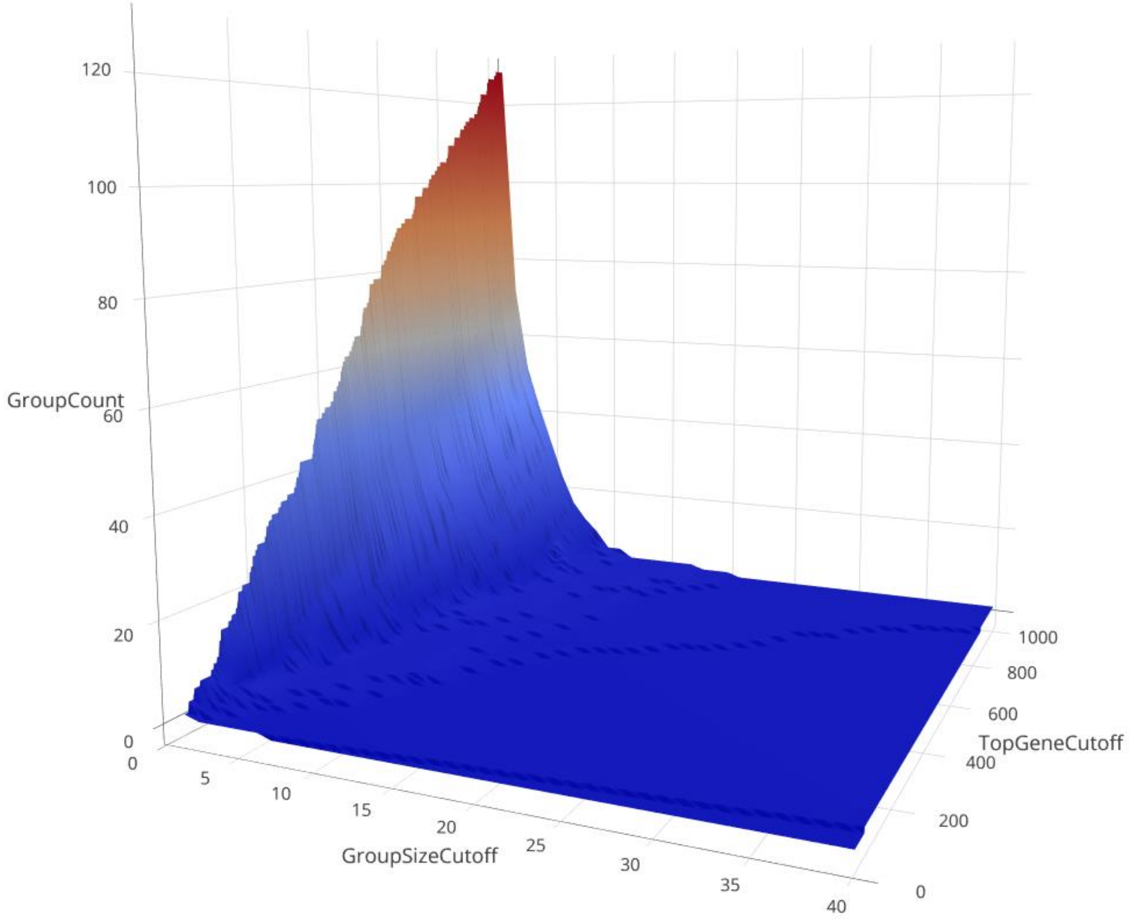


Eyecandy

2D Histogram + Scatter:



3D Surface:



Computational
Systems Biology

Thank you for your attention

Exercise cheatsheet