Plots.jl cheatsheet · Samuel S. Watson

Basics

1 Data are supplied to the `plot` function as arguments (`x`, or `x,y`, or `x,y,z`). Keyword arguments specify attributes.

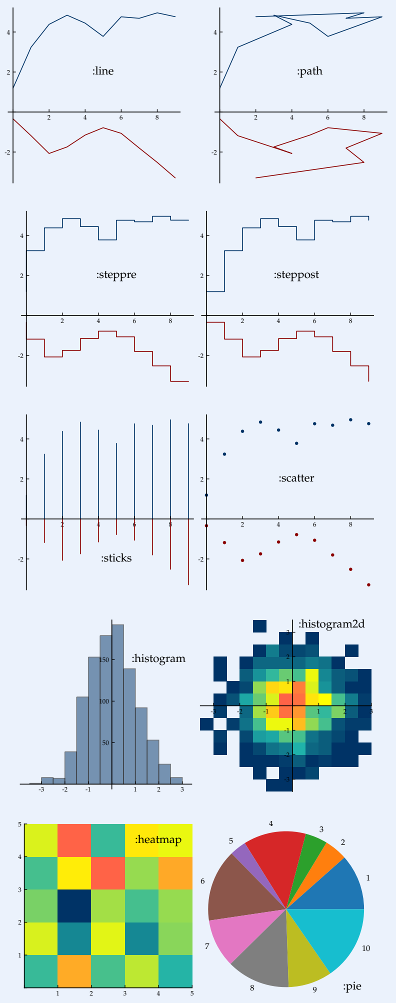
2 Arguments are interpreted flexibly: `x` and `y` can be vectors, or `x` can be a vector and `y` a function to be applied to x, or `x` can be omitted and inferred as `eachindex(y)`.

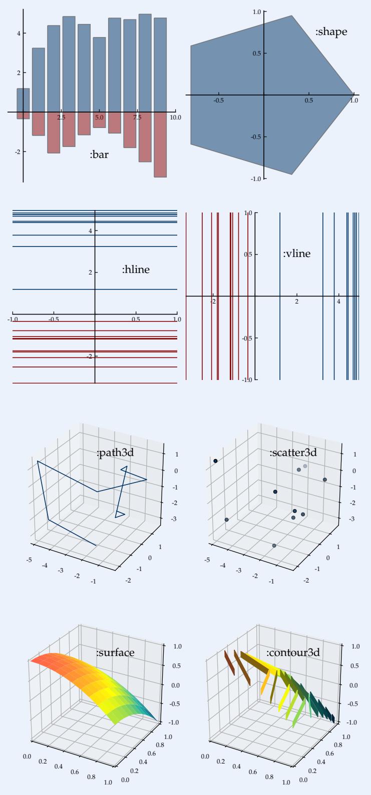
3 `plot(args...;kwargs...)` creates a new plot object, and `plot!(p,args...;kwargs...)` modifies the plot `p`. If omitted, `p` defaults to the plot `current()`.

4 A **series** is a set of data to be plotted together. The possible `seriestype`s are

```
[:line, :path, :steppre, :steppost, :sticks,
:scatter, :heatmap, :hexbin, :barbins, :barhist,
:histogram, :scatterbins, :scatterhist, :stepbins,
:stephist, :bins2d, :histogram2d, :histogram3d,
:density, :bar, :hline, :vline, :contour, :pie,
:shape, :image, :path3d, :scatter3d, :surface,
:wireframe, :contour3d, :volume]
```

The `seriestype` is specified as a keyword argument with key `seriestype` or `st`.





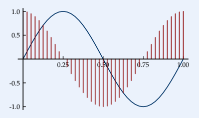
5 Most series types have function aliases, like `scatter(x,y)` for `plot(x,y,seriestype=:scatter)` and same for `scatter!`. Use the aliases for series docstrings (`?scatter`).

6 If a data argument or attribute is a 2D array, its **columns** are interpreted as separate series.

Combining plots

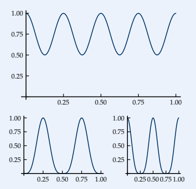
1 Series may be combined on the same axes using `plot!`.

```
x = 0:0.025:1
plot(x, x->sin(2π*x))
plot!(x, x->cos(2π*x),
    seriestype = :sticks)
```



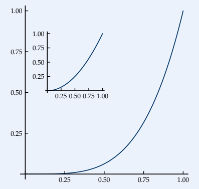
2 Series may be combined on separate axes using `@layout`.

```
l = @layout [a{0.6h};
             b{0.6w} c]
f(x) = sin(2π*x)^4 +
             cos(2π*x)^4
p1 = plot(x,f)
p2 = plot(x,x->sin(2π*x)^4)
p3 = plot(x,x->cos(2π*x)^4)
plot(p1, p2, p3, layout=l)
```



3 Inset plots: supply (parent plot index, bounding box) to `inset`. `bbox` arguments are `x, y, width, height`, each as a proportion of the corresponding parent plot dimension. Also, specify the `subplot` index for the new plot.

```
BB = bbox(0.15,0.15,
          0.35,0.35)
plot(x, x->x^4)
plot!(x, x->x^2,
      inset = (1, BB),
      subplot = 2)
```



Plot styling

1 **Plot attributes** (Default values followed by other possible values are shown in parentheses.)

(i) **Plots**

- `background_color/bg` (`RGB(1,1,1), :Firebrick`).
- `size` (`(600, 400)`, `(300, 300)`)
- `dpi` (`100`, `50`, `200`)
- `fontfamily` (`sans-serif`, `serif`)

(ii) **Subplots**

- `title` (`nothing`, `"My favorite plot"`)
- `legend/leg` (`:none`, `:best`, `:right`, `:left`, `:top`, `:bottom`, `:inside`, `:legend`, `:topright`, `:topleft`, `:bottomleft`, `:bottomright`)
- `framestyle/frame` (`:box`, `:semi`, `:axes`, `:origin`, `:zerolines`, `:grid`, `:none`)
- `aspect_ratio/ratio` (`:none`, `:equal`, `2.0`)
- `camera/cam` (`(30,30)`, `(45,45)`)
- `color_palette/palette` (`:auto`, `[:blue,:red,:green]`)

(iii) **Axes**

- `grid` (`true/false`)
- `gridlinewidth` (`0.5`, `0.25`, `1.0`)
- `gridstyle` (`:solid`, `:auto`, `:dash`, `:dot`)
- `link` (`:none`, `:x`, `:y`, `:both`, `:all`)
- `xlims`, `ylims`, `zlims`, (`:auto`, `(-10,5)`)
- `xticks`, `yticks`, `zticks` (`:auto`, `-4:2:4`)
- `xscale`, `yscale`, `yscale` (`:none`, `:ln`, `:log2`, `:log10`)
- `xguide/xlabel`, `yguide/ylabel` (`nothing`, `"time (s)"`)

2 **Series attributes**

(i) **Points**

- `markercolor/mc` (`:auto`, `:blue`, `RGB(0.2,0.4,0.2)`)
- `markeralpha/ma` (`1.0`, `0.5`, `0.2`)
- `markersize/ms` (`4`, `2`, `8`)
- `markershape/shape` (`:none`, `:auto`, `:circle`, `:rect`, `:star5`, `:diamond`, `:hexagon`, `:cross`, `:xcross`, `:utriangle`, `:dtriangle`, `:rtriangle`, `:ltriangle`, `:pentagon`, `:heptagon`, `:octagon`, `:star4`, `:star6`, `:star7`, `:star8`, `:vline`, `:hline`, `:+`, `:x`)
- `markerstrokecolor/msc` (`:auto`, `:blue`, `RGB(0,0,0)`)
- `markerstrokealpha/msa` (`1.0`, `0.5`, `0.2`)
- `markerstrokewidth/msw` (`0.5`, `1`)

(ii) **Lines**

- `linecolor/lc` (`:auto`, `:blue`, `RGB(0.2,0.4,0.2)`)
- `linealpha/la` (`1.0`, `0.5`, `0.2`)
- `linestyle/ls` (`:solid`, `:auto`, `:dash`, `:dot`, `:dashdot`, `:dashdotdot`)
- `linewidth/lw`

(iii) **Surfaces**

- `fillrange` (`nothing`, `0`, `sin.(x)`)
- `fillcolor/fc` (`:auto`, `:blue`, `RGB(0.2,0.4,0.2)`)
- `fillalpha/fa` (1.0, 0.5, 0.2)

Annotations and images

1 Add text with the `annotations/ann` attribute. Value should be a vector of tuples of the form `(x,y,txt)`, where `txt` is either a string or an object created with `text`.

```
ann = [(-π/2,-0.85,"min."),
       (-0.25,0.25,
       text("inflection point",
            pointsize=12, halign=:right,
            valign=:center, rotation=45))]
plot(sin, ann=ann)
# add arrowhead to line plot:
plot!([(-0.5,0.2),(-0.02,0.02)],arrow=1.0)
```

2 Add an image to a plot:

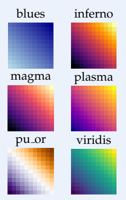
```
using Images
img = load("example.png")
x = range(-2, 2, length=size(img,1))
y = range(0, 1, length=size(img,2))
plot(x,y,img) # plots the image in [-2,2] × [0,1]
plot!(sin) # draw curve over image
```

Color gradients

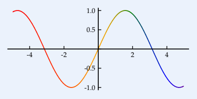
1 There are five collections of color gradients. `:Plots`, `:cmocean`, `:misc`, `:colorcet`, `:colorbrewer`. Choose one with `clibrary`.

2 Select your color gradient with `markercolor/linecolor/fillcolor`

3 Supply `z`-values for coloring with `marker_z/line_z/fill_z`



```
clibrary(:misc)
x = 0:0.01:1
plot(x, sin.(x),
     linecolor = :rainbow,
     line_z = cos.(x))
```



Miscellaneous

1 Data points can be grouped into separate series using the `group` attribute.

```
x,y = randn(100), randn(100)
class = rand(1:3, 100)
plot(x,y, group = class,
          color = [:blue :green :red])
```

2 `DataFrame` support:

```
using StatsPlots, DataFrames
D = DataFrame(a = randn(10),
              b = randn(10),
              c = rand(10))
@df D scatter(:a, :b, marker_z = :c)
```

3 **Recipes** provide support for custom types throughout Plots.

```
@recipe function f(A::Array{<:Complex})
    xguide := "Re(x)" # set attribute
    yguide --> "Im(x)" # set tentatively
    real.(A), imag.(A) # transformed data
end
```

4 `plotattr` provides information about plot attributes.

```
plotattr() # get help with plotattr
plotattr(:Series) # list Series attributes
plotattr("fill_z") # documentation for fill_z
```

5 Write figures to disk:

```
p = plot(x -> sin(x))
savefig(p, "myfig.pdf")
savefig("myfig.pdf") # uses p = current()
```

Formats for PyPlot backend are `eps`, `ps`, `pdf`, `png`, `svg`.