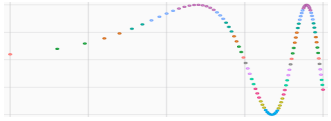
Gadfly Reference Card v0.2

(c) 2013 John Lynch – john.lynch@iname.com
Information taken liberally from the Gadfly documentation and various other sources. You may freely modify and distribute this document.



1 Introduction

Gadfly was developed by Daniel Jones to provide a system for plotting and visualization based on ggplot2 and the book "The Grammar of Graphics"
Details and examples can be found in the manual. http://dcjones.github.io/Gadfly.jl

Grammar represents an abstraction of objects aiming to shorten the distance from mind to page by mapping data to aesthetic attributes and geometric objects. Think of looking at how y changes with x across levels of z.

`Pkg.add("Gadfly")` to install
Load with `using Gadfly`

2 Invoking Plot .

This form is the standard "grammar of graphics" method of plotting. Data is supplied in the form of a dataframe, columns of the data are bound to *aesthetics*, and plot elements including *scales, coordinates, statistics, guides, and geometries* are added to the plot.

Conventional with dataframes

```
plot(data::AbstractDataFrame, elements::Element...; mapping...)
plot(dfname, x="c1", y="c2")                    generates plot of y vs x with points
plot(dfname, x="c1", y="c2", Geom.lines)        generates plot of y vs x with lines
plot(df ..., color="c3", ...point)              add colors based on categories in c3
```

Heretical with columns

```
plot(elements::Element...; mapping...)
plot(x=collect(1:20),y=fn(same length))         using arrays
```

Functions and Expressions

```
plot(f::Function,a,b, elements::Element...)
@plot(expr, a, b)
plot([sin, cos], 0, 10pi)                       Plot sin and cos of 0 to 10pi radians

set_default_plot_size(1cm,8cm)                  Set the default plot size
```

3 Modifying Aesthetics

```
plot(..., color ="c3", ...)                     color based on categories in c3
middle, lower_hinge, upper_hinge,               aesthetics for box plots
lower_fence, upper_fence, outliers
x_min, x_max, y_min, y_max                      aesthetics for rectbin
x, x_min, x_max, y, y_min, y_max                aesthetics for error bars
```

4 Geometries do the actual drawing

```
plot(..., Geom.point, ...)                      use points (default)
plot(..., Geom.lines, ...)                      use lines
plot(dfname, x="c1",  Geom.histogram)           use histogram
plot(..., Geom.bar, ...)
plot(..., Geom.bar, Geom.errorbar)
plot(...,  Geom.rectbin, color="c3")            use coloured 2d rectangles
plot(x=1:length(sds), y=ys, ymin=ymins,         plot error bars
ymax=ymaxs, Geom.point, Geom.errorbar)
plot(...color="c3", Geom.point,                 add loess smoothing of chart points
Geom.smooth)
plot(..., yintercept=[1.0, 2.0],                add horizontal line(s) to chart
     Geom.hline(color="red", size=3mm))
plot(..., xintercept=[1.0, 2.0], Geom.vline)    add vertical line(s) to chart
other variations                                see statistics
```

4 Statistics transform the one or more aesthetics

```
Stat.boxplot                                    outputs the middle, and upper and
plot(dfname, x="c1", y="c2", Geom.boxplot)      lower hinge, and upper and lower
                                                fence aesthetics
Stat.density                                    output line with kernel density
plot(df, x="price", Geom.density)               estimate from data
Geom.rectbin with Stat.histogram2d              heatmap style with density as colour
plot(dfname, x="c1", Geom.histogram2d)
Geom.line with Stat.smooth                      smoothing of data; method (loess);
plot(...color="c3", Geom.point, Geom.smooth)    smoothing=blahblah
```

5 Guides draw graphics to support the visualization such as axis ticks, labels and keys

```
plot(..., Guide.xlabel="Time")                  also ylabel
plot(..., Guide.colorkey("Key Title"))
```

6 Scales transform the data to aid visualization

```
Scale.x_log10   Scale.y_log10                   apply logarithmic scaling to axis
Scale.x_log2, Scale.x_log                       also to y axis
Scale.x_asinh, Scale.x_sqrt
```

7 Showing Facets

Facets are supported with subplot_grid

```
set_default_plot_size(20cm, 30cm)               Set plot size for multiple facets
plot(df,xgroup="c4", ygroup="c5", x="c1",       Plot grouped subplots
   y="c2", Geom.subplot_grid(Geom.point))
```

8 Output to other Formats

```
draw(SVG("myplot.svg", 6inch, 3inch), plot(...))    Also PDF and PS
draw(PNG("myplot.png", 6inch, 3inch), plot(...))
draw(D3("myplot.js", 15cm, 20cm), plot(...))
```

Using Gadfly to plot your Data

For a new session then you need to load plotting modules with **using Gadfly; using DataFrames**. You can enter it at the prompt or just include it at the top of a Julia file.

Load data from file if you need to
(the default separator is comma):
mydat = readtable("fname", separator='\t')
Check the data is right by looking at the first few rows
size(mydat) will show the number of rows, columns
mydat[1:3, 1: NumberOfColumns] for a sample

```
In [39]: data[1:5,1:5]
Out[39]:
5x5 DataFrame:
       randid   time1   time2 time2_1      sex
[1,]     2448    24.0 17.6263    24.0   "Male"
[2,]     6238    24.0    24.0    24.0 "Female"
[3,]     9428    24.0    24.0    24.0   "Male"
[4,]    10552 8.09309 8.09309     0.0 "Female"
[5,]    11252    24.0    24.0 11.7317 "Female"
```

Then lets look at the data.
First lets do a histogram of the data in column 3:
plot(mydat[3], Geom.histogram)

Perhaps a boxplot to get a statistical view:
plot(y=mydat[3], Geom.boxplot)
If its too narrow or wide then change the plot size:
set_default_plot_size(6cm, 10cm)
and then plot it again

If you to check the main statistics then
println(mean(mydat[3], " ", std(mydat[2]), " ",m+1.96*sd)
describe(mydat[2])

N