2014-04-23.sagews

April 23, 2014

Contents

1 Math 480b Sage Course

1.1 More Graphics

1.2 April 23, 2014

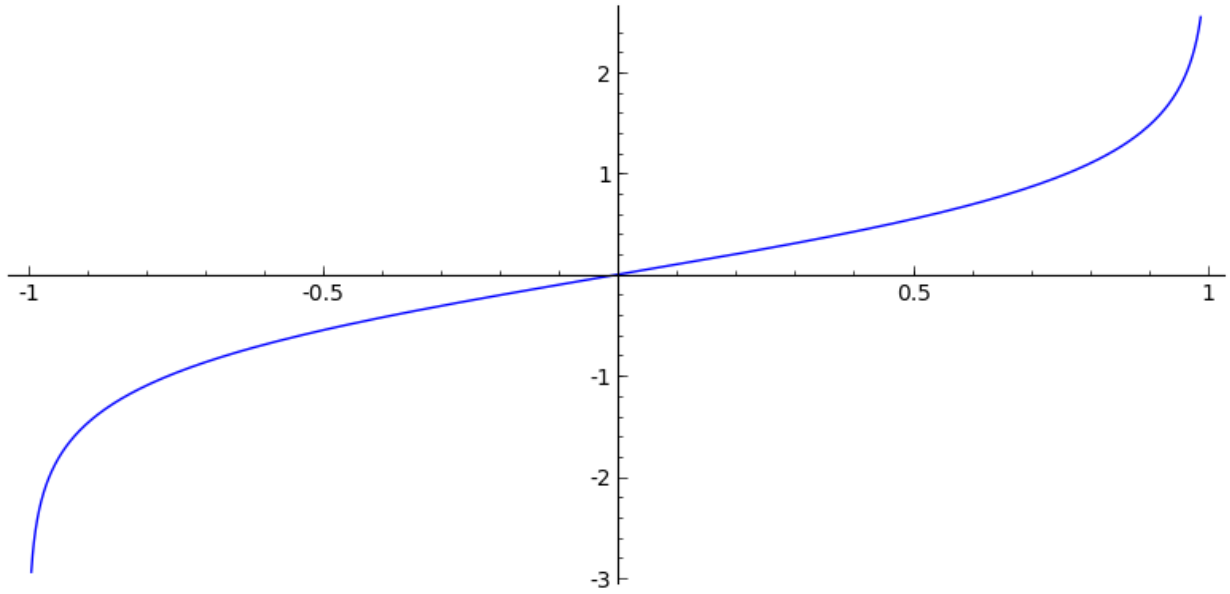
Screencast: `http://youtu.be/Q9NitcAoRb4`

Plan

- Questions?
- Including a plot in a LaTeX document: step-by-step tutorial
- 2D matplotlib graphics in Sage
- 3D graphics in Sage
- NEXT topic (starting Friday): git, the command line terminal, sage development. (Well delay Cython.)

1.3 Including a plot in a LaTeX document: step-by-step tutorial

```
g = plot(arctanh)
g
```




```
g.save('arctanh.pdf')
```

1.4 2d Graphics using Matplotlib (like Matlab, but better)

- matplotlib is an easy-to-install standard Python plotting library (which Sage uses extensively).
- examples/docs at the gallery: `http://matplotlib.org/gallery.html`
- how to get them to appear in SageMathCloud worksheets

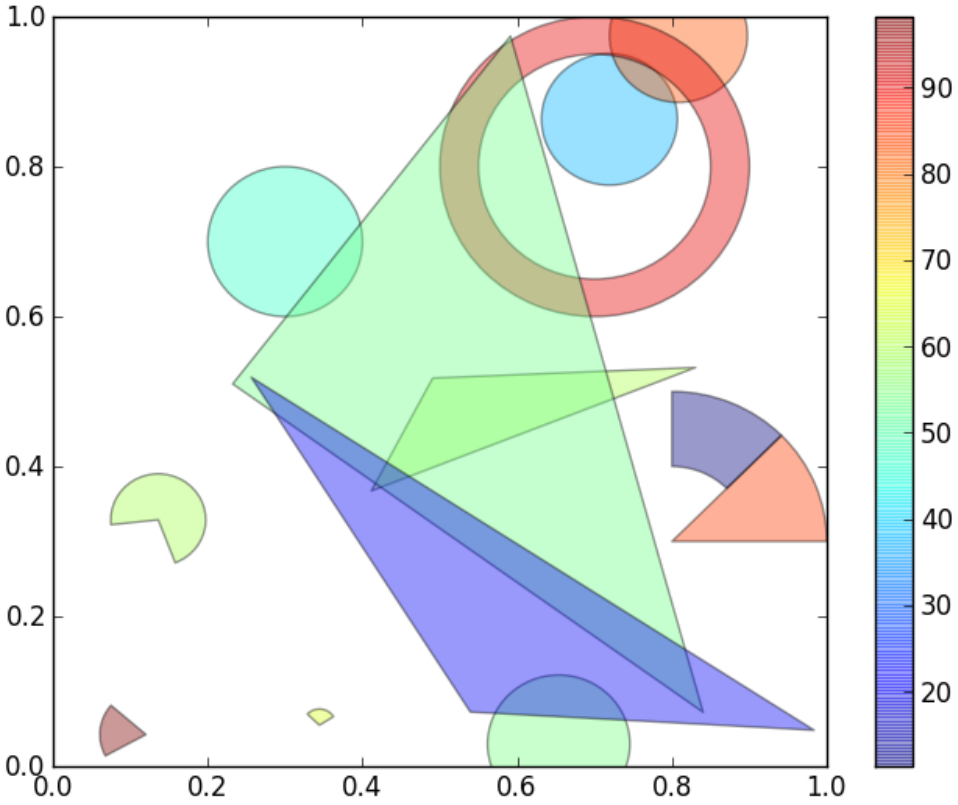
```
v = [3,4,7]]
w = [1,9,18]
[(3,1), (4,9), (7,18)]
[(v[i],w[i]) for i in range(len(v))]
zip(v,w)
```

```
[(3, 1), (4, 9), (7, 18)]
[(3, 1), (4, 9), (7, 18)]
[(3, 1), (4, 9), (7, 18)]
```

```
import numpy as np
import matplotlib
from matplotlib.patches import Circle, Wedge, Polygon
from matplotlib.collections import PatchCollection
import matplotlib.pyplot as plt


fig, ax = plt.subplots()

resolution = 50 # the number of vertices
N = 3
x       = np.random.rand(N)
y       = np.random.rand(N)
```

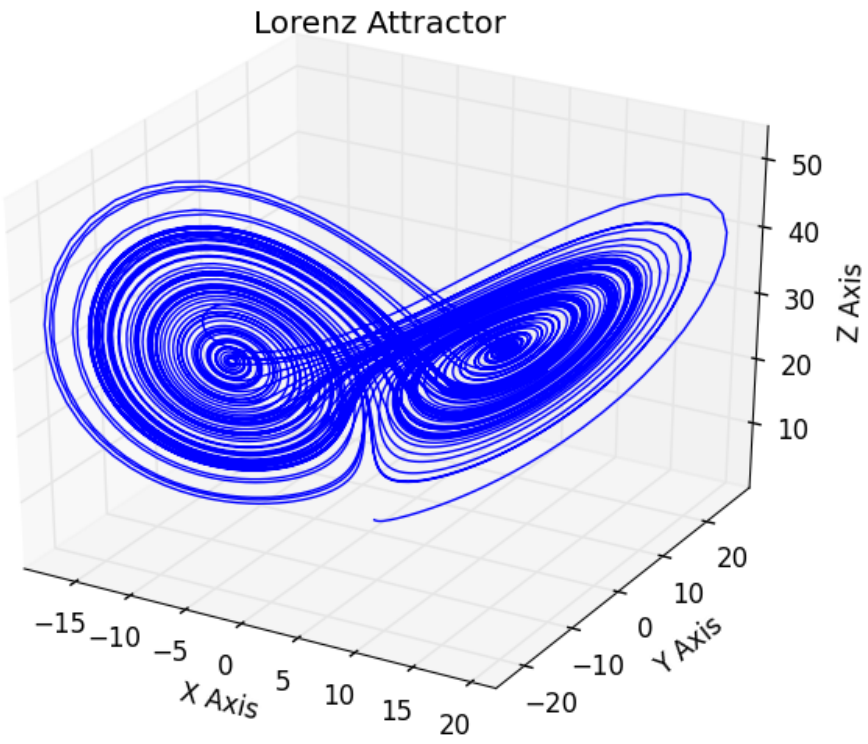
```
radii   = 0.1*np.random.rand(N)
patches = []
for x1,y1,r in zip(x, y, radii):
    circle = Circle((x1,y1), r)
    patches.append(circle)

x       = np.random.rand(N)
y       = np.random.rand(N)
radii   = 0.1*np.random.rand(N)
theta1  = 360.0*np.random.rand(N)
theta2  = 360.0*np.random.rand(N)
for x1,y1,r,t1,t2 in zip(x, y, radii, theta1, theta2):
    wedge = Wedge((x1,y1), r, t1, t2)
    patches.append(wedge)

# Some limiting conditions on Wedge
patches += [
    Wedge((.3,.7), .1, 0, 360),             # Full circle
    Wedge((.7,.8), .2, 0, 360, width=0.05), # Full ring
    Wedge((.8,.3), .2, 0, 45),              # Full sector
    Wedge((.8,.3), .2, 45, 90, width=0.10), # Ring sector
]

for i in range(N):
    polygon = Polygon(np.random.rand(N,2), True)
    patches.append(polygon)

colors = 100*np.random.rand(len(patches))
p = PatchCollection(patches, cmap=matplotlib.cm.jet, alpha=0.4)
p.set_array(np.array(colors))
ax.add_collection(p)
plt.colorbar(p)

plt.show()
```

```
<matplotlib.collections.PatchCollection object at 0xafbb650>
<matplotlib.colorbar.Colorbar instance at 0xa79f758>
```




```
# from http://matplotlib.org/examples/mplot3d/lorenz_attractor.html

import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D


def lorenz(x, y, z, s=10, r=28, b=2.667) :
    x_dot = s*(y - x)
    y_dot = r*x - y - x*z
    z_dot = x*y - b*z
    return x_dot, y_dot, z_dot


dt = 0.01
stepCnt = 10000

# Need one more for the initial values
xs = np.empty((stepCnt + 1,))
ys = np.empty((stepCnt + 1,))
zs = np.empty((stepCnt + 1,))
```

```
# Setting initial values
xs[0], ys[0], zs[0] = (0., 1., 1.05)

# Stepping through "time".
for i in range(stepCnt) :
    # Derivatives of the X, Y, Z state
    x_dot, y_dot, z_dot = lorenz(xs[i], ys[i], zs[i])
    xs[i + 1] = xs[i] + (x_dot * dt)
    ys[i + 1] = ys[i] + (y_dot * dt)
    zs[i + 1] = zs[i] + (z_dot * dt)

fig = plt.figure()
ax = fig.gca(projection='3d')

_ = ax.plot(xs, ys, zs)
_ = ax.set_xlabel("X Axis"), ax.set_ylabel("Y Axis"), ax.set_zlabel("Z \
   Axis"), ax.set_title("Lorenz Attractor")

plt.show()
```



1.5 3d Sage Graphics (like Mathematica)

- function of two variables (surface)
- regular polyhedra
- sphere
- implicit plot of surface
- text

```
f(x,y) = x+y^2*sin(x*y)
g = plot3d(f, (x, -5,2), (y,-1,1), color='purple')
g += plot3d(x-y, (x, -5, 2), (y,-1,1), color='red')
g
```

```
f(x,y) = x+y
g = plot3d(f, (x, -1,1), (y,-1,1), color='red')
```

```
sphere(opacity=.5, color='green', mesh=2) + icosahedron(color='red', \
   opacity=.7, size=2).translate((1,0,0))
```

```
T = RDF(golden_ratio)
%var x,y,z
p = 2 - (cos(x + T*y) + cos(x - T*y) + cos(y + T*z) + cos(y - T*z) + cos(\
   z - T*x) + cos(z + T*x))
r = 4.77
implicit_plot3d(p, (x, -r, r), (y, -r, r), (z, -r, r), plot_points=50, \
   color='orange')
```

```
reset('r')
```

```
%r
cars <- c(1,3,6,4,9)
```

```
%r
cars
```

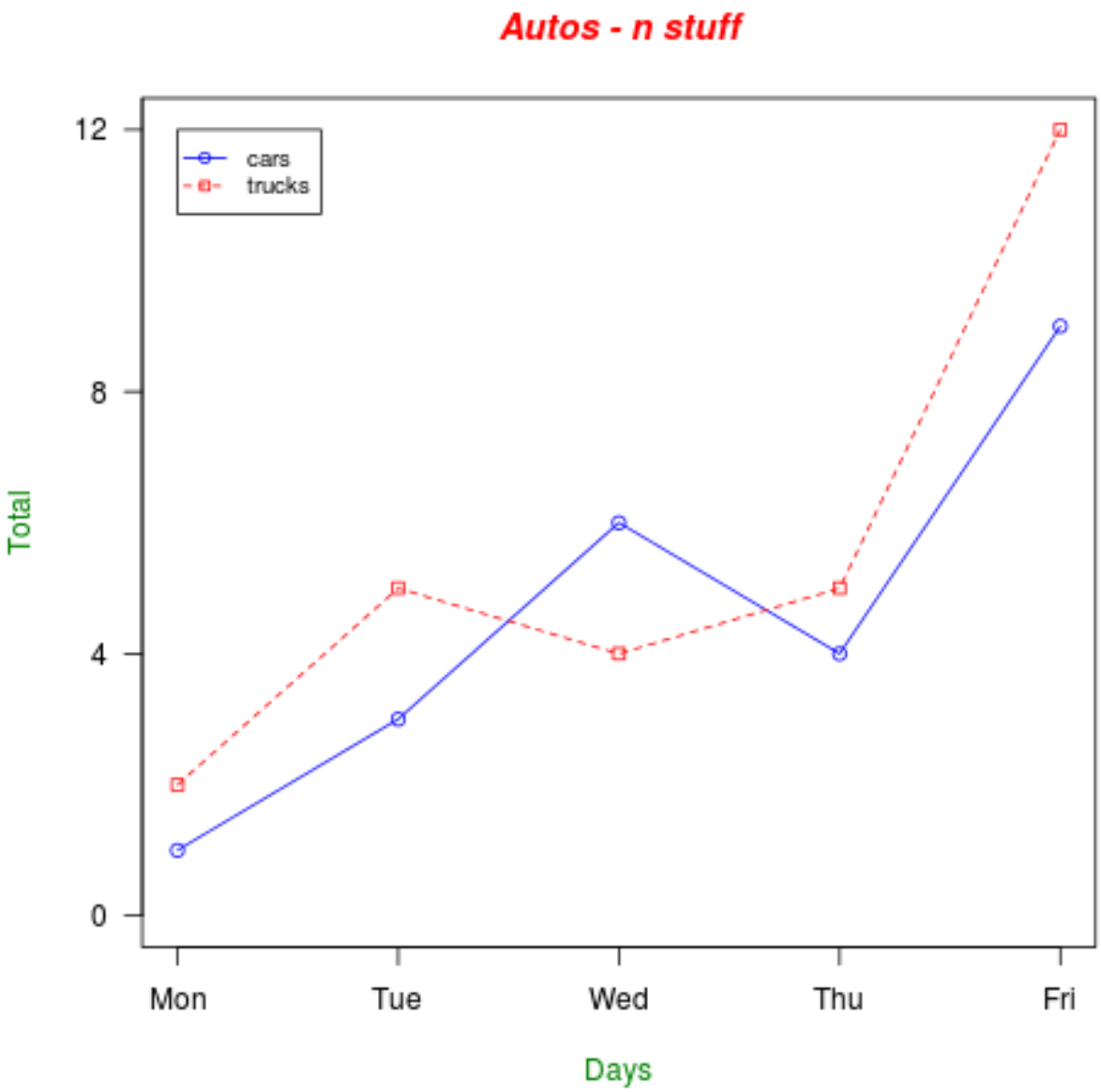
```
[1] 1 3 6 4 9
```

1.6 Other You can also draw plots using R

To draw a plot (or do anything) using R, put

```
%r

# Define 2 vectors
cars <- c(1, 3, 6, 4, 9)
trucks <- c(2, 5, 4, 5, 12)

# Calculate range from 0 to max value of cars and trucks
g_range <- range(0, cars, trucks)
```

```
# Graph autos using y axis that ranges from 0 to max
# value in cars or trucks vector.  Turn off axes and
# annotations (axis labels) so we can specify them ourself
plot(cars, type="o", col="blue", ylim=g_range,
   axes=FALSE, ann=FALSE)

# Make x axis using Mon-Fri labels
axis(1, at=1:5, lab=c("Mon","Tue","Wed","Thu","Fri"))

# Make y axis with horizontal labels that display ticks at
# every 4 marks. 4*0:g_range[2] is equivalent to c(0,4,8,12).
axis(2, las=1, at=4*0:g_range[2])

# Create box around plot
box()

# Graph trucks with red dashed line and square points
lines(trucks, type="o", pch=22, lty=2, col="red")

# Create a title with a red, bold/italic font
title(main="Autos - n stuff", col.main="red", font.main=4)

# Label the x and y axes with dark green text
title(xlab="Days", col.lab=rgb(0,0.5,0))
title(ylab="Total", col.lab=rgb(0,0.5,0))

# Create a legend at (1, g_range[2]) that is slightly smaller
# (cex) and uses the same line colors and points used by
# the actual plots
legend(1, g_range[2], c("cars","trucks"), cex=0.8,
   col=c("blue","red"), pch=21:22, lty=1:2);
```



1.7 Or even plot with octave

```
%octave
cd('/tmp')
M = rand(10);
h = figure('visible', 'off');
plot(M);
saveas(h,"figure2.png");
```

```
salvus.file('/tmp/figure2.png')
```