math 480 - 20080523 - intro stats

Math 480: 20080523

TODAY

1. Hand in graded homework
2. Discuss projects and schedule for the next two weeks
3. Scipy.stats -- fast intro
4. Student feedback and evaluations.

Final Projects: Discussion

1. Your final project counts for **40 percent of your grade**. There is no final exam.
2. Projects will all be posted on the wiki at the end of the course.
3. Projects are due Friday, June 6, 2008.

Schedule

1. Next Wednesday: You get feedback back on your projects
2. Next Wed/Friday -- stats (numpy/scipy and R)
3. Week before finals -- three days of project presentations

Presentations

1. We have 150 minutes total, and about 20 projects, so 7 minutes/each. No "going over time" will be permitted.
2. Each project *must* "register" with me before final week. Handaround paper today.

3. I will draw up a schedule with similar project presentations grouped together, and hand it out next week.
4. 20% of your final project grade will come from student feedback on your presentation. During the presentations each student will score your presentation/project on a scale of 1 to 3, based on whatever subjective methods they want. These will be averaged and count for 20% of your final project grade. (Caveat: If any student scores seem out of whack I'll ignore them.) It is thus very important that you attend every day.
5. 20% of your final project grade will come from the TA (Robert Miller)
6. The remaining 60% will be a subjective score by me based on:
   1. Your presentation.
   2. Overall Quality (e.g., do things well versus terribly)
   3. Interest (to me, other students).
   4. Whether or not you finish what you attempt to do.
   5. A prototype of something that is in no shape to get into Sage can result in a perfect grade if it illustrates something really cool.
   6. If you are unsure, ASK!

Statistics -- scipy.stats (just some quick remarks)

The webpage about scipy.stats is here:

http://www.scipy.org/SciPyPackages/Stats

The docs:

1. Continuous Distributions [pdf]
2. Discrete Distributions [pdf]

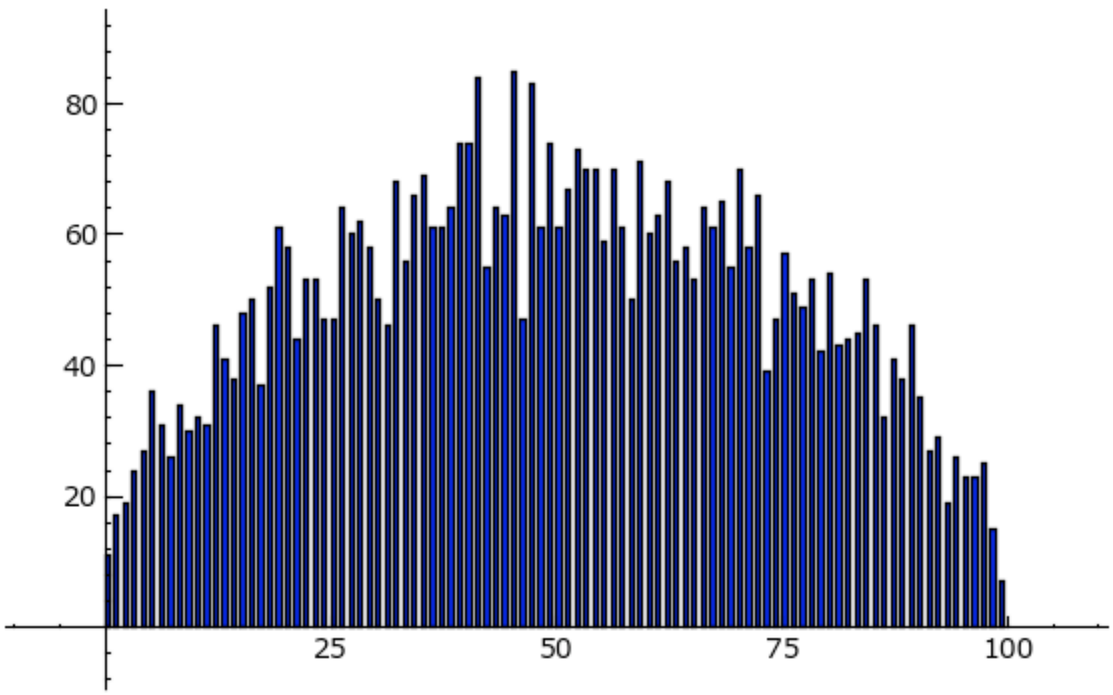
```
import scipy
import scipy.stats
```

```
A = scipy.stats.semicircular()
```

```
time v = A.rvs(5000)
v[:5]
```

```
array([-0.30108917,  0.41080692, -0.0695175 ,  0.88366313,
0.52467243])
CPU time: 10.51 s,  Wall time: 10.60 s
```

```
bar_chart(scipy.histogram(v,100)[0])
```



WARNING: scipy.stats definitely has some SERIOUS performance issues

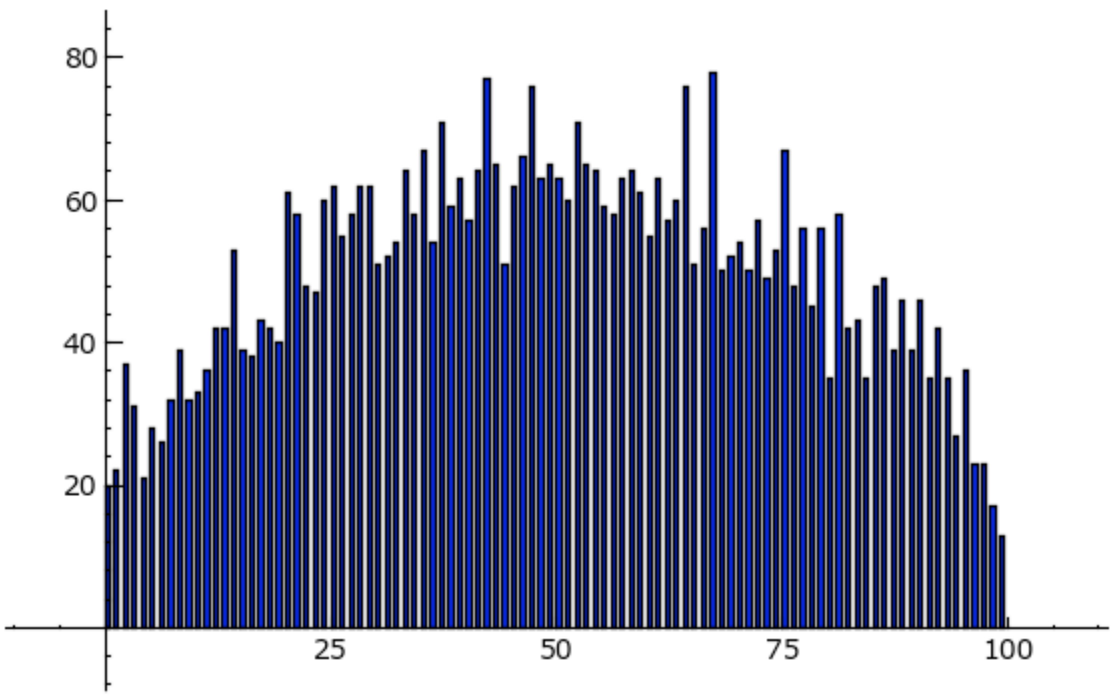
The source code for a lot of scipy.stats is in pure Python. Some of it is very slow, evidently. E.g., just by solving numerically I was able to write something easily that is much faster than scipy at choosing numbers from a semicircle distribution (see below). So watch out if you're trying to do large Monte-Carlo experiments...

```
f = (2/pi) * ( integrate(sqrt(1-x^2)) + pi/4)
g  = f.expand()
ff = f._fast_float_(x)
def g(t):
   return sage.numerical.optimize.find_root(lambda x: ff(x)-t,
-1.0r,1.0r)
```

```
time w = [g(random()) for _ in range(5000)]
```

```
    CPU time: 0.15 s,  Wall time: 0.15 s
```

```
bar_chart(scipy.histogram(w,100)[0])
```




```
scipy.stats.mean(w)
```

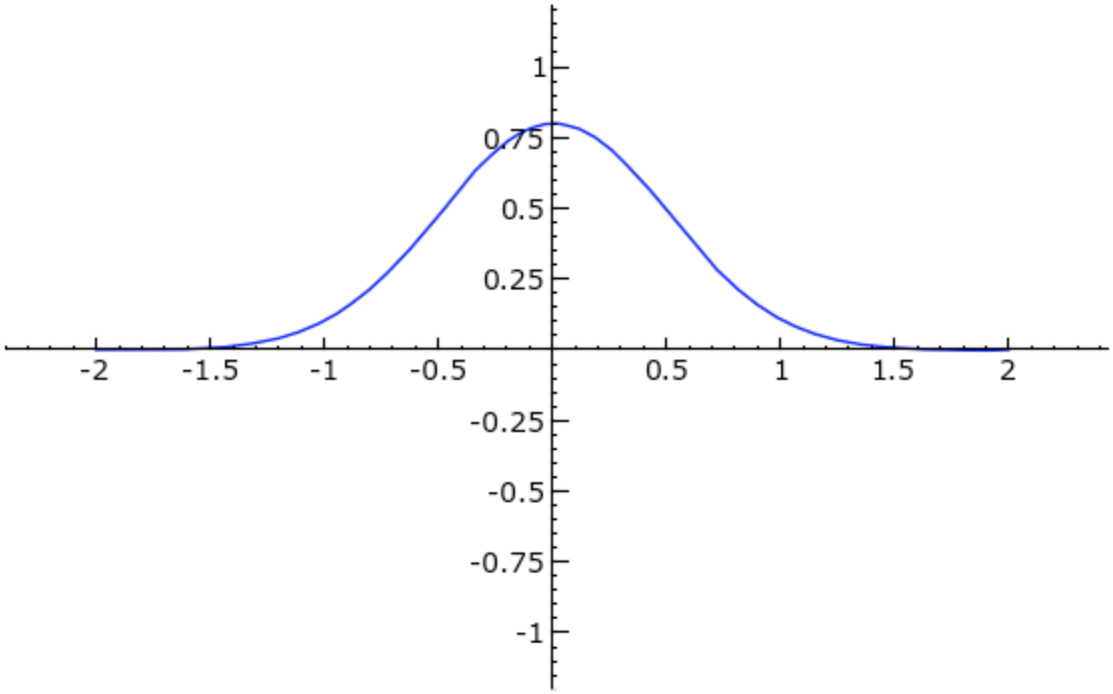
```
    0.006754909851979787
```

```
scipy.stats.std(w)
```

```
    0.50049757157867669
```

```
mu = scipy.stats.norm(scipy.stats.mean(w), scipy.stats.std(w))
```

```
plot(mu.pdf,-2,2)
```



Summary: Sage has a ton of statistical capabilities, and also includes R, which adds a lot more. But I think that much much more work is needed since (1) using R from Sage is quite awkward and unpythonic, and (2) scipy.stats has 80 continuous distributions and 10 discrete ones, and they are very easy to use; however, there are major performance issues.

Emily Kirkman will be working fulltime this summer on this. If you're also very interested, contact me.