Lab Class 1: Heavy tails and the Noah Effect

Nick Watkins

May 15, 2012

1 Probability distributions

These instructions assume you have some familiarity with running Matlab, editing scripts etc. Please talk to me or the other people on hand, if you need help with this. I am also assuming you'll save the commands in a script file that begins with:

```
close all
clear all
```

in order to reset the state of the workspace when you run them.

1.1 Measures of dependence

First make some "white" (independent), identically distributed, Gaussian noise, with pdf given by

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-(x-\mu)/\sigma\right)^2 \tag{1}$$

using the

```
normrnd
```

command in the stats toolbox.

```
MU=0
SIGMA=1
noise=normrnd(MU,SIGMA,1000,1);
```

and plot it:

```
figure(1);plot(noise)
title('white Gaussian noise')
xlabel('time')
ylabel('amplitude')
```

Then add up the noise to make a (Brownian) random walk:

```
walk=cumsum(noise);
figure(2);plot(walk)
title('Brownian random walk')
xlabel('time')
ylabel('amplitude')
```

To demonstrate that the noise is indeed Gaussian we can look at its pdf. First step here is to use a histogram:

```
figure(3);hist(noise)
xlabel('amplitude')
ylabel('counts')
```

Exercise: Adapt above to make a histogram of the random walk. Is this Gaussian ? Why ? What do you see if you replace normrnd with another distribution such as the exponential ?

Exercise: A histogram is not a pdf. Why not ? How would you adapt the above code to make it a pdf. What other diagnostics might you use ?

1.2 Heavy tailed distributions

Now want to do same thing but with noise drawn from a heavy-tailed distribution. Choose the generalised Pareto distribution, for which Matlab has a function in the stats toolbox.

$$p(x) = \frac{1}{\sigma}\left(1 + k\frac{x-\theta}{\sigma}\right)^{-(1+1/k)} \tag{2}$$

We run it in the limit when $k > 0$ and $\theta = \sigma/k$, when it is equivalent to the Pareto distribution.

```
K=1
SIGMA=1
THETA=SIGMA/K
heavynoise=gprnd(K,SIGMA,THETA,1000,1);

figure(4);plot(heavynoise)
title('white Pareto noise')
xlabel('time')
ylabel('amplitude')
```

and again generate a walk from it:

```
heavywalk=cumsum(heavynoise);
figure(5);plot(heavywalk)
title('Heavy-tailed walk')
xlabel('time')
ylabel('amplitude')
```

Now take a first look at this distribution with a histogram:

```
figure(6);hist(heavynoise,200)
xlabel('amplitude')
ylabel('counts')
```

where I have taken 200 bins. For the reasons discussed in Cosma's second lecture a power law is not very obvious in such a plot.

Another way to take a quick look at the data is a log log plot of the histogram:

```
figure(7);
[x,n]=hist(heavynoise,200);
loglog(n,x,'*')
xlabel('amplitude')
ylabel('counts')
```

Exercise: The above plot has a number of disadvantages that may not be at first obvious. In particular, note that we are still using uniformly sized bins. Why is this not a good idea for a power law ? How would you change the code to compensate ? How would you further modify it to calculate a pdf rather than a histogram ?

An interesting alternative is a rank order plot, where we plot each statistic against its rank:

```
figure(8)
ranked=sort(heavynoise,'descend');
loglog(ranked,1:1000,'*')
grid on
title(strcat('Rank order plot for 1000 Pareto random variables,  K=',num2str(K)))
ylabel('rank')
xlabel('statistic')
```

To get a better feel for what this is telling you try different distributions, e.g. lognormal and exponential. The rank order plot is closely related to the complementary cumulative distribution function $1 - P(x)$, where:

$$P(x) = \int_{x_0}^{x} p(x')dx' \tag{3}$$

To get a more accurate handle on estimation of a power law tail, you'll want to explore for yourself the use of the Clauset et al codes, the basis for which is discussed in Cosma's lectures and the Clauset, Shalizi and Newman paper in SIAM Review **51(4)**, 661-703 (2009). For serious use always check for updates and other information at

`http://tuvalu.santafe.edu/~aaronc/powerlaws/`

As an example to get you started we will use some of their routines to examine the heavy-tailed noise:

```
[alpha, xmin, L]=plfit(heavynoise)
plplot(heavynoise,xmin,alpha);
title('Maximum likelihood estimate of best power law tail, plotted over ccdf')
```

Exercise: If you didn't know before can you now see how the rank order plot is related to the cdf ?

How does the power law exponent returned compare to what you'd expect for the Pareto ? What does x_{min} try to estimate ?

Suggestions for further study: A couple of things you can try on your own are:

i) generate plots of the running means and running standard deviations from the Gaussian, exponential, Pareto data used above. What do these plots suggest to you about the problem of insuring against different types of risk ?

ii) Simulate a noise drawn from a lognormal distribution. Use the various methods you have looked at so far to explore the statistics of the lognormal data. How would you distinguish a lognormal tail from a power law tail ?