Power-law distributions

From InterSciWiki

see also Zipf's law

Maximal likelihood estimation of the parameters of different distributions as compared to those of the power law is an extremely important and delicate matter. The software discussed here, developed for a review article by Clauset, Shalizi, and Newman, offer many advantages over methods of estimation currently in use. The latter methods often given biased results, especially for smaller sample. MLE methods give asymptotically unbiased estimats as sample size increases. Alternative models that can be tested with this software include log-normal, exponential, stretched exponential, and power law with a cutoff in the continous and discrete cases, and Poisson in the discrete case. ~~~~ Doug White 07:19, 9 September 2007 (PDT) (First instructional edition, DRW and LT)

Contents

MLE (Maximal Likelihood Estimation) Estimation, Error Bars, and Model Comparison

Here is the [R and Matlab software and documentation (http://www.santafe.edu/~aaronc/powerlaws/)] (Updated 29 June 2007) for the 2007 review article, "**Power-law distributions in empirical data** (http://arxiv.org/abs/0706.1062) ," by Aaron Clauset, Cosma Rohilla Shalizi, and M. E. J. Newman. See also Power laws, Pareto distributions and Zipf's law (http://arxiv.org/abs/cond-mat/0412004) Contemporary Physics 46, 323-351 (2005)

Abstract: Power-law distributions occur in many situations of scientific interest and have significant consequences for our understanding of natural and man-made phenomena. Unfortunately, the empirical

detection and characterization of power laws is made difficult by the large fluctuations that occur in the tail of the distribution. In particular, standard methods such as least-squares fitting are known to produce systematically biased estimates of parameters for power-law distributions and should not be used in most circumstances. Here we describe statistical techniques for making accurate parameter estimates for power-law data, based on maximum likelihood methods and the Kolmogorov-Smirnov statistic. We also show how to tell whether the data follow a power-law distribution at all, defining quantitative measures that indicate when the power law is a reasonable fit to the data and when it is not. We demonstrate these methods by applying them to twenty-four real-world data sets from a range of different disciplines. Each of the data sets has been conjectured previously to follow a power-law distribution. In some cases we find these conjectures to be consistent with the data while in others the power law is ruled out.

Contents of the companion files: Matlab and R programs

concise on-line guide to using MATLAB (http://web.cecs.pdx.edu/~gerry/MATLAB/)

- Random number generators - randht.m (Matlab; used in plvar.m below to generate an empirical vector generated according to alpha and Xmin)

(Always check original at SFI for current updates of randht.m!)

Fitting a power-law distribution (Matlab)

copy/paste plfit.m in the work folder, then:

- plfit.m (Matlab) - cut and paste your data over (1-rand (10000,1)).^(-1/(2.5-1)) in the format of the example in the code, which is

```
x = (1-rand (10000,1)).^(-1/(2.5-1));
    [alpha, xmin, D] = plfit(x);
```

Input: e.g., (this is a command; its execution does not change array x)

```
x = (5400,4800, 4000, ...etc);
    [alpha, xmin, D] = plfit(x);
```

once the randht.m function is loaded into Matlab, paste the command into the command window and run.

Output:

alpha (PL slope) and Xmin.

(Always check original at SFI for current updates of plfit.m!)

Estimating uncertainty in the fitted parameters (Matlab)

copy/paste plvar.m in the work folder, then:

- plvar.m (Matlab) Updated 25 July 2007.

Input: as above, e.g., (this is a command; its execution does not change array x)

```
x = (5400,4800, 4000, ...etc);
[alpha, xmin, ntail] = plvar(x);
```

Output:

D = Kolmogorov-Smirnov D = difference between actual data and true PL with these parameters, 0 = perfect fit.
s.e of Alpha (model alpha also given) Alpha=the slope of the PL in the PL region
s.e of Xmin (model Xmin also given) Xmin=the lower cutoff at which the PL no longer applies
s.e of nTail (model nTail also given) nTail=the number of observations in the PL region

(Always check original at SFI for current updates of plvar.m!)

- plpva.m (Matlab; uses randht.m).

Calculating p-value for fitted power-law model (Matlab)

copy/paste plpva.m in the work folder, then:

Input: as above, e.g., (this is a command; its execution does not change array x)

```
x = (5400,4800, 4000, ...etc);
[[p, gof] = plpva(x, 1);
```

Output:

p-value and gof.

Checking the power-law fits with a graph (Matlab)

plver.m Verify fits (http://intersci.ss.uci.edu/wiki/pw/plver.m) from Verify fits (download Matlab code: Aaron Clauset (http://intersci.ss.uci.edu/wiki/pw/plverify.m))

```
x = [500,150,90,81,75,75,70,65,60,58,49,47,40]
put in the work folder
plfit.m
plvar.m
plpva.m
plver.m
```

Input: as above, e.g., (this is a command; its execution does not change array x)

```
x = [500,150,90,81,75,75,70,65,60,58,49,47,40];
```

```
[alpha, xmin, D] = plfit(x);
[a_err, xm_err, nt_err] = plvar(x);
[p, gof] = plpva(x, xmin);
x = reshape(x,numel(x),1);
q = unique(x); n = length(x);
c = hist(x,q)'./n;
c = [[q; q(end)+1] 1-[0; cumsum(c)]]; c(find(c(:,2)<10^-10),:) = [];
cf = ([xmin:q(end)]'.^-alpha)./(zeta(alpha) - sum([1:xmin-1].^-alpha));
cf = [[xmin:q(end)+1]' 1-[0; cumsum(cf)]];
cf(:,2) = cf(:,2) .* c(find(c(:,1)==xmin),2);
figure(1);
loglog(c(:,1),c(:,2),'bo','MarkerSize',8,'MarkerFaceColor',[1 1 1]); hold on;
loglog(cf(:,1),cf(:,2),'k--','LineWidth',2); hold off;
set(gca,'FontName','Times','FontSize',16);
```

Output:

an eps graph for the power-law fit.

(Always check original at SFI for current updates of plver.m - plverify.m is not yet posted at the Clauset site)

Calculating likelihood-ratio test results (R)

This is done in statistical programming language R (http://www.r-project.org/) .

Documentation (http://www.santafe.edu/~aaronc/powerlaws/Rcode_README.txt)

(Always check original at SFI for current updates!)

Comparing fit of different models using LR=Loglikelihood Ratios

Shalizi R code (http://www.santafe.edu/~aaronc/powerlaws) (note: this is not yet a proper R package). is at

R code (http://www.santafe.edu/~aaronc/powerlaws/pk_R-v0.3-2007-07-25.tgr) in a downloadable tgz file Updated 25 July 29 July 2007
Zip file (http://intersci.ss.uci.edu/wiki/pw/pli-R-v0.0.3-2007-07-25.zip) Updated 25 July 29 July 2007
Documentation (http://www.santafe.edu/~aaronc/powerlaws/Rcode_README.txt)

Comparing Pareto v. Lognormal fit. Inside R copy and paste (or your modifications)

```
source(power-law-test.R)
source(pareto.R)
source( lnorm.R)
x = c(500,150,90,81,75,75,70,65,60,58,49,47,40)
```

ppareto(x, threshold=1, exponent=2.7, lower.tail=TRUE, log.p=FALSE)
which helps to remember the settings, and can be in any order; or just do
ppareto(x, 1, 2.7, TRUE, FALSE)

But for these city size data Cosma thinks we shouldn't calculate the cumulative distribution function at each of your data points. "What you want to do instead is create a fitted Pareto distribution, which would be a command

```
x.pareto <- pareto.fit(x,threshold=1)
drw: better fit!
```

or whatever threshold you decide to use. Of course it doesn't have to be called "x.pareto", but I found that convention a useful mnemonic. Similarly, after loading the lnorm.R functions --- making sure you use the same threshold for both fits, you'll want to do

```
x.lnorm <- lnorm.fit(x,threshold=1)
drw: error, so I adjusted to
x.lnorm <- lnorm.fit(x,threshold=40)
x.pareto <- pareto.fit(x,threshold=40)
```

or, for the cutoff of 47

```
x.lnorm <- lnorm.fit(x,threshold=47)
x.pareto <- pareto.fit(x,threshold=47)
```

Then you can do

```
vuong(pareto.lnorm.llr(x,x.pareto,x.lnorm))
```

The functions pareto.fit and lnorm.fit produce R data structures, and the log-likelihood ratio functions, like pareto.lnorm.llr use those structures to extract the necessary parameters."

```
drw: cool. Now I get
$loglike.ratio
[1] 20.28356
$mean.LLR
[1] 1.560274
$sd.LLR
[1] 0.8133283
$Vuong
[1] 6.916822
$p.one.sided
[1] 1
$p.two.sided
[1] 4.618972e-12
drw: So the fit is better for lognormal than Pareto. Very nice!
```

(Always check original at SFI for current updates!)

Comparing fit of Tsallis q-exponential using LR=Loglikelihood Ratios

Shalizi R code (http://www.santafe.edu/~aaronc/powerlaws) (note: this is not yet a proper R package). is at

R code (http://www.santafe.edu/~aaronc/powerlaws/pk_R-v0.3-2007-07-25.tgr) in a downloadable tgz file Updated 25 July 29 July 2007
Zip file (http://intersci.ss.uci.edu/wiki/pw/pli-R-v0.0.3-2007-07-25.zip) Updated 25 July 29 July 2007
Documentation (http://www.santafe.edu/~aaronc/powerlaws/Rcode_README.txt)

Comparing Pareto v. Lognormal fit. Inside R copy and paste (or your modifications)

source(power-law-test.Tsal.R)
source(pareto.R)
#source(lnorm.R)
source(tsal.R)
#c900
x = c(500,150,90,81,75,75,70,65,60,58,49,47,40)
ppareto(x, threshold=1, exponent=2.7, lower.tail=TRUE, log.p=FALSE)
#which helps to remember the settings, and can be in any order; or just do
ppareto(x, 1, 2.7, TRUE, FALSE)

But for these city size data Cosma thinks we shouldn't calculate the cumulative distribution function at each of your data points. "What you want to do instead is create a fitted Pareto distribution, which would be a command

```
x.pareto <- pareto.fit(x,threshold=1)
drw: better fit!
```

or whatever threshold you decide to use. Of course it doesn't have to be called "x.pareto", but I found that convention a useful mnemonic. Similarly, after loading the lnorm.R functions --- making sure you use the same threshold for both fits, you'll want to do

x.lnorm <- lnorm.fit(x,threshold=1) DRW:ltsal.fit is inside tsal.R

```
x.tsal  <-  tsal.fit(x,threshold=47)
x.pareto <- pareto.fit(x,threshold=47)
```

Then you can do

vuong(pareto.Tsal.llr(x,x.pareto,x.tsal))

```
source(zeta.R)
vuong(zeta.Tsal.llr(x,x.pareto,x.tsal))
```

The functions pareto.fit and ltsal.fit produce R data structures, and the log-likelihood ratio functions, like pareto.ltsal.llr use those structures to extract the necessary parameters." So then

```
$loglike.ratio
```

```
drw: So the fit is better for ..... than Pareto.
```

(Always check original at SFI for current updates!)

Pareto.R

Inside R copy and paste (or your modifications) with appropriate tresholds from Matlab for batch runs

```
source(pareto.R)
#c900
x = c(500,150,90,81,75,75,70,65,60,58,49,47,40)
x.pareto <- pareto.fit(x,threshold=47)
x.pareto
#c1000
x = c(400,90,85,80,78,70,68,58,50,50,50,50,45,43,40,40,40,40)
x.pareto <- pareto.fit(x,threshold=43)
x.pareto
#c1100
x = c(442,100,90,88,85,80,67,63,55,55,50,48,45,45,45,43,40,40,40)
x.pareto <- pareto.fit(x,threshold=40)
x.pareto
#c1150
x = c(150,145,130,100,100,90,80,72,72,60,60,56,52,45,45)
x.pareto <- pareto.fit(x,threshold=60)
x.pareto
#c1200
x = c(255,130,130,110,100,100,80,80,74,70,67,63,60,50,50,50,50,47,46)
x.pareto <- pareto.fit(x,threshold=47)
x.pareto
#c1250
x = c(320,140,140,130,100,100,90,80,77,75,67,60,60,54,50,47)
x.pareto <- pareto.fit(x,threshold=67)
x.pareto
#c1300
x = c(432,401,150,118,95,91,90,85,84,80,70,54,50,50,50,48)
x.pareto <- pareto.fit(x,threshold=48)
x.pareto
#c1350
x = c(432,400,150,114,96,95,90,90,90,75,70,50,50,50)
x.pareto <- pareto.fit(x,threshold=70)
x.pareto
#c1400
x = c(487,235,150,150,129,115,90,90,81,75,65,60,60,60,51,50,50)
```

```
x.pareto <- pareto.fit(x,threshold=50)
x.pareto
```

TOP 10 Cities

```
#c900
x = c(500,150,90,81,75,75,70,65,60,58,49,47,40)
x.pareto <- pareto.fit(x,threshold=58)
x.pareto
#c1000
x = c(400,90,85,80,78,70,68,58,50,50,50,50,45,43,40,40,40,40)
x.pareto <- pareto.fit(x,threshold=50)
x.pareto
#c1100
x = c(442,100,90,88,85,80,67,63,55,55,50,48,45,45,45,43,40,40,40)
x.pareto <- pareto.fit(x,threshold=55)
x.pareto
#c1150
x = c(150,145,130,100,100,90,80,72,72,60,60,56,52,45,45)
x.pareto <- pareto.fit(x,threshold=60)
x.pareto
#c1200
x = c(255,130,130,110,100,100,80,80,74,70,67,63,60,50,50,50,50,47,46)
x.pareto <- pareto.fit(x,threshold=70)
x.pareto
#c1250
x = c(320,140,140,130,100,100,90,80,77,75,67,60,60,54,50,47)
x.pareto <- pareto.fit(x,threshold=75)
x.pareto
#c1300
x = c(432,401,150,118,95,91,90,85,84,80,70,54,50,50,50,48)
x.pareto <- pareto.fit(x,threshold=80)
x.pareto
#c1350
x = c(432,400,150,114,96,95,90,90,90,75,70,50,50,50)
x.pareto <- pareto.fit(x,threshold=75)
x.pareto
#c1400
x = c(487,235,150,150,129,115,90,90,81,75,65,60,60,60,51,50,50)
x.pareto <- pareto.fit(x,threshold=75)
x.pareto
#c1450
x = c(...)
x.pareto <- pareto.fit(x,threshold=.....)
x.pareto
#c1450
```

```
x = c(...)
x.pareto <- pareto.fit(x,threshold=.....)
x.pareto
```

1. --> graph algorithm from Matlab

```
#c1500
x = c(...)
x.pareto <- pareto.fit(x,threshold=.....)
x.pareto

#c1550
x = c(...)
x.pareto <- pareto.fit(x,threshold=.....)
x.pareto
```

Update status

7 September 2007: corrected interim reporting in plpva.m; changed plfit.m, plvar.m and plpva.m to reshape input vector to column format, and to prevent using continuous approximation in small-sample regime for discrete data.

Comparing Tsallis q to other distributions

In progress: Tsallis q distribution project Clauset, Tambayong, White

Categories: Matlab software | R software | Q exponential | Distributions | Major contributors