

Internet Measurement and Data Analysis (5)

Kenjiro Cho

2012-10-31

review of previous class

Class 4 Distribution and confidence intervals (10/24)

- ▶ Normal distribution
- ▶ Confidence intervals and statistical tests
- ▶ Distribution generation
- ▶ exercise: confidence intervals
- ▶ **assignment 1**

today's topics

Class 5 Diversity and complexity

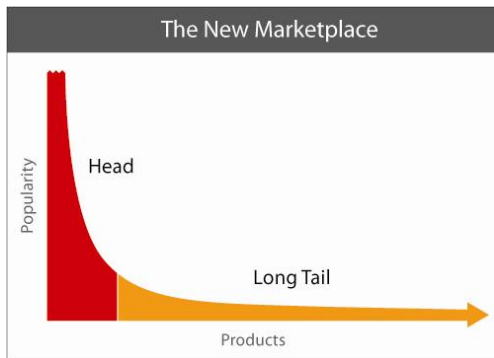
- ▶ Long tail
- ▶ Web access and content distribution
- ▶ Power-law and complex systems
- ▶ exercise: power-law analysis

long tail

a business model for online retail services

- ▶ head: a small number of bestseller items: for real stores
- ▶ tail: diverse low-sales items: covered by online stores

it is now widely used for diverse niche market



source: <http://longtail.com/>

complex systems

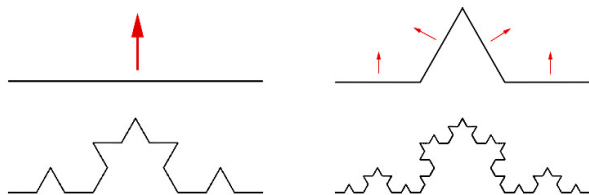
complex systems science

- ▶ a system with interfering components that as a whole exhibits complex behavior not obvious from the individual components
- ▶ the real world is full of complex systems
- ▶ difficult to analyze by traditional methods based on reductionism
 - ▶ need to understand a complex system as is, without decomposition
- ▶ many studies started in 1990's
 - ▶ few remaining problems that can be solved with reductionism
 - ▶ analysis and simulations enabled by computers

power-law and complex systems

power-law

- ▶ one of the characteristics of complex systems
 - ▶ power-law: observed variable changes in proportion to a power of some parameter
 - ▶ self-similarity (fractal)
- ▶ observed in various natural and social phenomena and Internet services
- ▶ scale-free: no typical scale



Koch curve: fractal image similar to coastline

Zipf's law

- ▶ an empirical law formulated in 1930's about frequency in ranked data
- ▶ the share is inversely proportional to its rank
 - ▶ the share of the k th ranked item is proportional to $1/k$
- ▶ observed in social science, natural science and data communications
 - ▶ the frequency of English words, the population of cities, wealth distribution, etc
 - ▶ file size, network traffic
- ▶ long-tail in a linear-scale plot, heavy-tail in a log-log plot

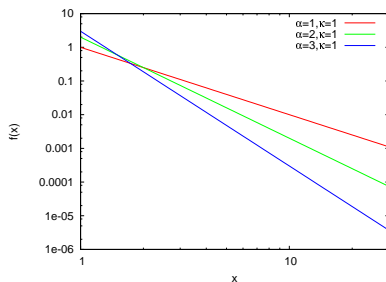
power-law distribution

- ▶ power-law distribution: the probability of observing a value is proportional to a power of the value

$$f(x) = ax^k$$

- ▶ appears as a straight-line in a log-log plot

$$\log f(x) = k \log x + \log a$$



complexity of the Internet

complexity of topology (network science)

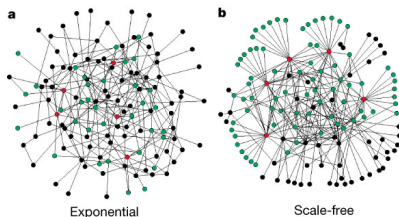
- ▶ scale-free: the degree distribution of nodes follows a power-law
 - ▶ many small-degree nodes and a small number of large-degree nodes
 - ▶ highest-degree nodes greatly exceed the average degree
- ▶ small-world:
 - ▶ compact: the average distance between 2 nodes is short
 - ▶ clusters: nodes are highly clustered

traffic behavior (time-series analysis)

- ▶ self-similarity
- ▶ long-range dependence

scale-free network

- ▶ the degree distribution of network nodes follows power-law
 - ▶ many small-degree nodes, small number of large-degree nodes
 - ▶ highest-degree nodes greatly exceed the average degree
- ▶ small-world
 - ▶ compact: the average distance between 2 nodes is short
 - ▶ clusters: nodes are highly clustered
- ▶ construction: preferential attachment: rich get richer
 - ▶ higher probability to attach to a high-degree node
- ▶ fault-tolerance, attack-tolerance
 - ▶ robust against random failures
 - ▶ vulnerable to an attack to a hub node



source: Error and attack tolerance of complex networks. R. Albert, H. Jeong, A. Barabasi. Nature 406, July 2000.

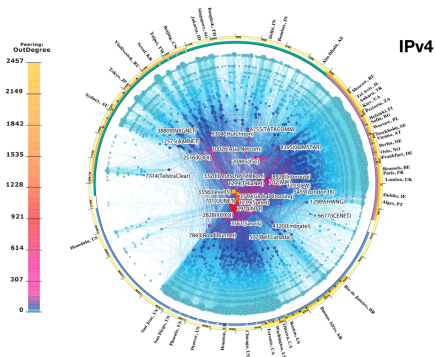
example: AS structure of the Internet

CAIDA AS CORE MAP 2009/03

- ▶ visualization of AS topology using skitter/ark data
- ▶ longitude of AS (registered location), out-degree of AS

IPv4
INTERNET TOPOLOGY MAP
JANUARY 2009

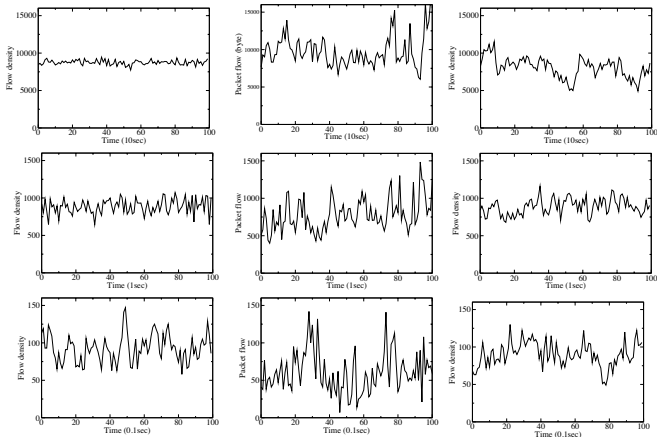
AS-level INTERNET GRAPH



copyright © 2009 UC Regents. all rights reserved.

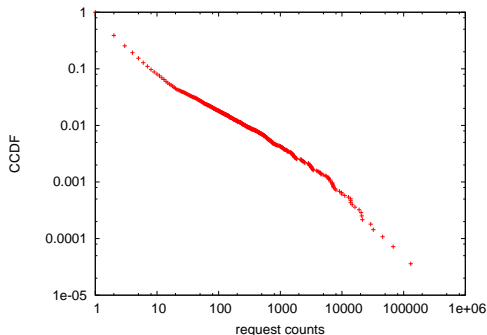
self-similarity in network traffic

- ▶ exponential model (left), real traffic (middle), self-similar model (right)
- ▶ time scale: 10sec (top), 1 sec (middle), 0.1 sec (bottom)



Web access and content distribution

- ▶ power-law can be observed everywhere on the web
 - ▶ the number of incoming links and access count of web page, occurrences of search keywords



content access count distribution of the JAIST web server

various distributions

- ▶ binomial distribution
- ▶ poisson distribution
- ▶ normal distribution
- ▶ exponential distribution
- ▶ power-law distribution

binomial distribution

- ▶ bernoulli trial: a trial is random and has only 2 outcomes
- ▶ discrete probability distribution of the number of success k for n trials, with the probability of success p for a trial

PDF

$$P[X = k] = \binom{n}{k} p^k (1 - p)^{n-k}$$

here

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

$$\text{mean : } E[X] = np, \text{ variance : } \text{Var}[X] = np(1 - p)$$

when n is large, a binomial distribution can be approximated by a poisson distribution

Poisson distribution

the occurrence rate of rare events follows Poisson distribution

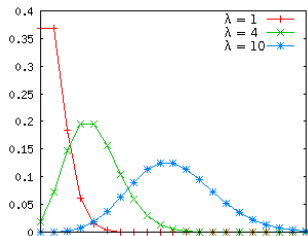
- ▶ death toll of traffic accidents, the number of mutations of DNA, etc

Poisson distribution is expressed by a single expected value $\lambda > 0$

PDF

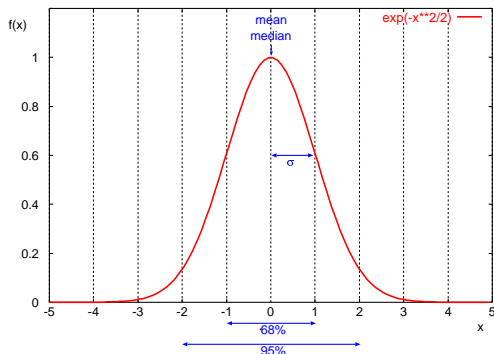
$$P(X = k) = \frac{\lambda^k e^{-\lambda}}{k!}$$

mean : $E[X] = \lambda$, variance : $Var[X] = \lambda$



normal distribution (1/2)

- ▶ also known as gaussian distribution
- ▶ defined by 2 parameters: μ :mean, σ^2 :variance
- ▶ sum of random variables follows normal distribution
- ▶ standard normal distribution: $\mu = 0, \sigma = 1$
- ▶ in normal distribution
 - ▶ 68% within (*mean - stddev, mean + stddev*)
 - ▶ 95% within (*mean - 2 * stddev, mean + 2 * stddev*)



normal distribution (2/2)

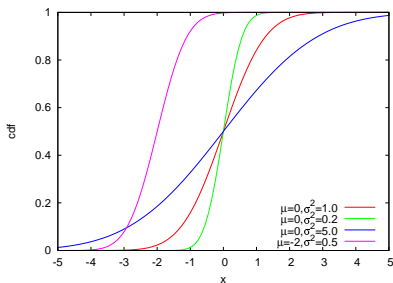
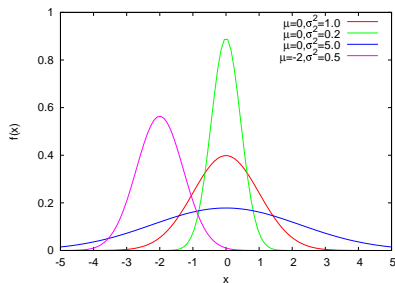
probability density function (PDF)

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

cumulative distribution function (CDF)

$$F(x) = \frac{1}{2} \left(1 + \operatorname{erf} \frac{x - \mu}{\sigma\sqrt{2}} \right)$$

μ : mean, σ^2 : variance



exponential distribution

the intervals of independent events occurring at a constant rate follow an exponential distribution

- ▶ call intervals in telephone systems, session intervals of TCP connections, etc

PDF

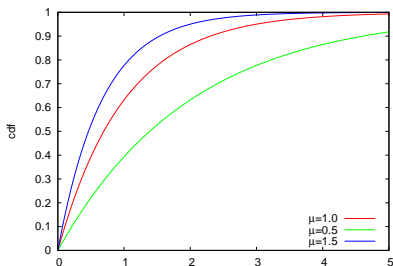
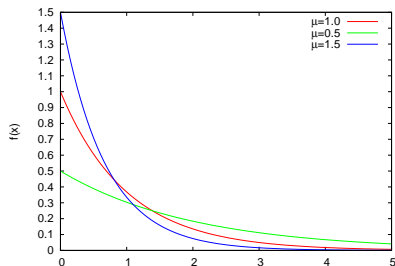
$$f(x) = \lambda e^{-\lambda x}, (x \geq 0)$$

CDF

$$F(x) = 1 - e^{-\lambda x}$$

$\lambda > 0$: rate parameter

mean : $E[X] = 1/\lambda$, variance : $\text{Var}[X] = \lambda^{-2}$



pareto distribution

most widely used power-law distribution in networking research

PDF

$$f(x) = \frac{\alpha}{\kappa} \left(\frac{\kappa}{x}\right)^{\alpha+1}, (x > \kappa, \alpha > 0)$$

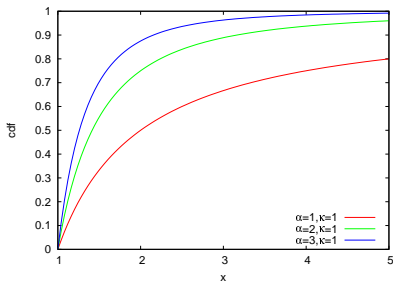
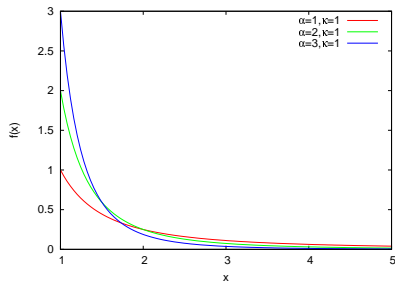
CDF

$$F(x) = 1 - \left(\frac{\kappa}{x}\right)^{\alpha}$$

κ : minimum value of x , α : pareto index

$$\text{mean} : E[X] = \frac{\alpha}{\alpha - 1} \kappa, (\alpha > 1)$$

if $\alpha \leq 2$, variance $\rightarrow \infty$. if $\alpha \leq 1$, mean and variance $\rightarrow \infty$.



CCDF

Complementary Cumulative Distribution Function (CCDF)
in power-law distribution, the tail of distribution is often of interest

ccdf: probability of observing x or more

$$F(x) = 1 - P[X \leq x]$$

- ▶ plot ccdf in log-log scale
 - ▶ to see the tail of the distribution or scaling property

plotting CCDF

to plot CDF

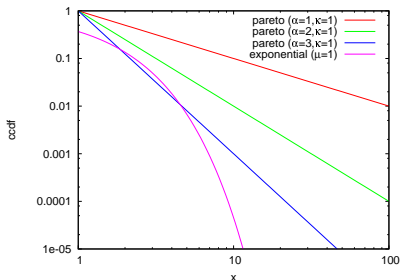
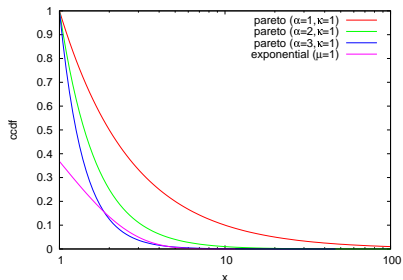
- ▶ sort $x_i, i \in \{1, \dots, n\}$ by value
- ▶ plot $(x_i, \frac{1}{n} \sum_{k=1}^i k)$
- ▶ Y-axis is usually in linear scale

to plot CCDF

- ▶ sort $x_i, i \in \{1, \dots, n\}$ by value
- ▶ plot $(x_i, 1 - \frac{1}{n} \sum_{k=1}^i k)$
- ▶ both X-axis and Y-axis are in log scale

CCDF of pareto distribution

- ▶ log-linear (left)
 - ▶ exponential distribution: straight line
- ▶ log-log (right)
 - ▶ pareto distribution: straight line



previous exercise: generating normally distributed random numbers

- ▶ generating pseudo random numbers that follow the normal distribution
 - ▶ write a program to generate normally distributed random numbers with mean μ and standard deviation σ , using a uniform random number generator function (e.g., `rand` in ruby)
- ▶ plotting a histogram
 - ▶ generate random numbers that follow the standard normal distribution, plot the histogram to confirm the standard normal distribution.
- ▶ computing confidence intervals
 - ▶ observe confidence interval changes according to sample size use the normally distributed random number generator to produce 10 sets of normally distributed random numbers with mean 60 and standard deviation 10. sample size $n = 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048$
 - ▶ compute the confidence interval of the population mean from each sample set
use confidence level 95% and confidence interval " $\pm 1.960 \frac{\sigma}{\sqrt{n}}$ ".
plot the results of 10 sets in a single graph. plot sample size n on the X-axis in log-scale and mean and confidence interval on the Y-axis in linear scale

box-muller transform

basic form: creates 2 normally distributed random variables, z_0 and z_1 , from 2 uniformly distributed random variables, u_0 and u_1 , in $(0, 1]$

$$z_0 = R \cos(\theta) = \sqrt{-2 \ln u_0} \cos(2\pi u_1)$$

$$z_1 = R \sin(\theta) = \sqrt{-2 \ln u_0} \sin(2\pi u_1)$$

polar form: approximation without trigonometric functions
 u_0 and u_1 : uniformly distributed random variables in $[-1, 1]$,
 $s = u_0^2 + u_1^2$ (if $s = 0$ or $s \geq 1$, re-select u_0, u_1)

$$z_0 = u_0 \sqrt{\frac{-2 \ln s}{s}}$$

$$z_1 = u_1 \sqrt{\frac{-2 \ln s}{s}}$$

random number generator code by box-muller transform

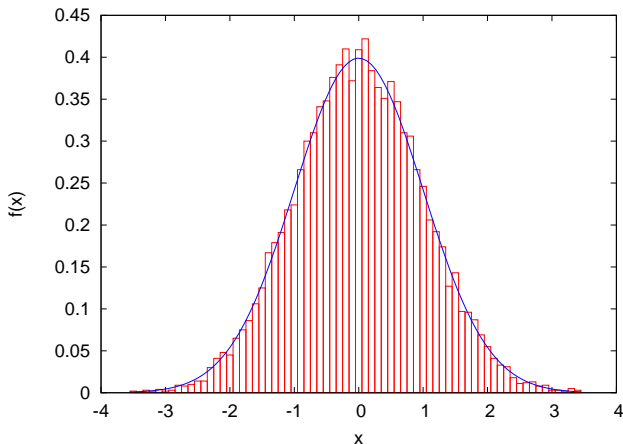
```
# usage: box-muller.rb [n [m [s]]]
n = 1 # number of samples to output
mean = 0.0
stddev = 1.0

n = ARGV[0].to_i if ARGV.length >= 1
mean = ARGV[1].to_i if ARGV.length >= 2
stddev = ARGV[2].to_i if ARGV.length >= 3

# function box_muller implements the polar form of the box muller method,
# and returns 2 pseudo random numbers from standard normal distribution
def box_muller
  begin
    u1 = 2.0 * rand - 1.0 # uniformly distributed random numbers
    u2 = 2.0 * rand - 1.0 # ditto
    s = u1*u1 + u2*u2 # variance
    end while s == 0.0 || s >= 1.0
    w = Math.sqrt(-2.0 * Math.log(s) / s) # weight
    g1 = u1 * w # normally distributed random number
    g2 = u2 * w # ditto
    return g1, g2
  end
# box_muller returns 2 random numbers. so, use them for odd/even rounds
x = x2 = nil
n.times do
  if x2 == nil
    x, x2 = box_muller
  else
    x = x2
    x2 = nil
  end
  x = mean + x * stddev # scale with mean and stddev
  printf "%.6f\n", x
end
```

plot a histogram of normally distributed random numbers

- ▶ plot a histogram of random numbers following the standard normal distribution, and confirm that they are normally distributed
- ▶ generate 10,000 random numbers from the standard normal distribution, use bins with one decimal place for the histogram



plotting a histogram

- ▶ plot a histogram using bins with one decimal place

```
#
# create histogram: bins with 1 digit after the decimal point
#

re = /(-?\d*\.\d+)/ # regular expression for input numbers

bins = Hash.new(0)

ARGF.each_line do |line|
  if re.match(line)
    v = $1.to_f
    # round off to a value with 1 digit after the decimal point
    offset = 0.5 # for round off
    offset = -offset if v < 0.0
    v = Float(Integer(v * 10 + offset)) / 10
    bins[v] += 1 # increment the corresponding bin
  end
end

bins.sort{|a, b| a[0] <=> b[0]}.each do |key, value|
  puts "#{key} #{value}"
end
```

plotting a histogram of the standard normal distribution

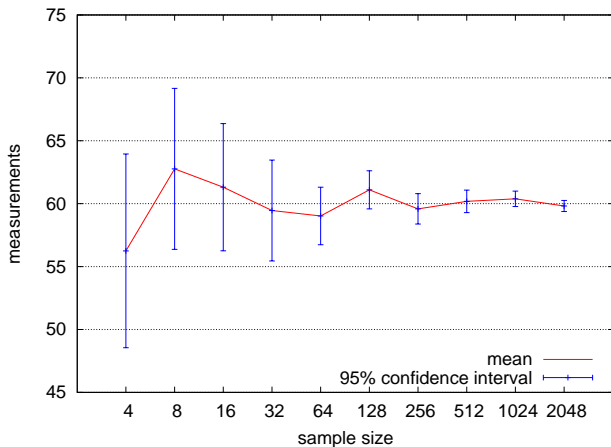
```
set boxwidth 0.1
set xlabel "x"
set ylabel "f(x)"
plot "box-muller-histogram.txt" using 1:($2/1000) with boxes notitle, \
      1/sqrt(2*pi)*exp(-x**2/2) notitle with lines linetype 3
```

note: probability density function (PDF) of standard normal distribution

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$$

the confidence interval of sample mean and sample size

the confidence interval becomes narrower as the sample size increases



the confidence interval of sample mean and sample size

assignment 1: the finish time distribution of a marathon

- ▶ purpose: investigate the distribution of a real-world data set
- ▶ data: the finish time records from honolulu marathon 2010
 - ▶ <http://results.sportstats.ca/res2010/honolulu.htm>
 - ▶ the number of finishers: 20,181
- ▶ items to submit
 1. mean, standard deviation and median of the total finishers, male finishers, and female finishers
 2. the distributions of finish time for each group (total, men, and women)
 - ▶ plot 3 histograms for 3 groups
 - ▶ use 10 minutes for the bin size
 - ▶ use the same scale for the axes to compare the 3 plots
 3. CDF plot of the finish time distributions of the 3 groups
 - ▶ plot 3 groups in a single graph
 4. optional
 - ▶ other analysis of your choice (e.g., CDF plots of age groups or countries)
 5. discussion
 - ▶ describe your observations about the data and plots
- ▶ submission format: a single PDF file including item 1-5
- ▶ submission method: upload the PDF file through SFC-SFS
- ▶ submission due: 2012-11-09

honolulu marathon data set

data format

Place	Chip Time	Pace /mi	#	Name	City	ST	CNT	Gender Plce/Tot	Category Plc/Tot	@10km Category	@21.1 Split1	@30 Split2	
1	02:15:18	5:10	4	Chelimo, Nicholas	Ngong Hills	KEN	1/10586	1/9	MELite	32:57	1:07:41	1:3	
2	02:17:18	5:15	3	Limo, Richard	Eldoret	KEN	2/10586	2/9	MELite	32:56	1:07:40	1:3	
3	02:19:54	5:21	5	Bushendich, Solomon	Eldoret	KEN	3/10586	3/9	MELite	32:57	1:07:40	1:3	
4	02:20:58	5:23	8	Kirwa, Gibert	Iten	KEN	4/10586	4/9	MELite	32:56	1:07:40	1:3	
5	02:22:34	5:27	1	Muindi, Jimmy	Kangundo	KEN	5/10586	5/9	MELite	32:56	1:08:11	1:3	
6	02:22:36	5:27	2	Hussein, Mbarak	Albuquerque	NM	USA	6/10586	6/9	MELite	32:57	1:09:57	1:4
7	02:27:25	5:38	11	Stanko, Nicholas	Haslette	MI	USA	7/10586	7/9	MELite	32:57	1:10:22	1:4
8	02:30:20	5:45	29712	Ogura, Makoto	Hiroshima	Hi	JPN	8/10586	1/1229	M35-39	34:11	1:13:14	1:4
9	02:32:13	5:49	9670	Gebre, Belainesh	Flagstaff	AZ	USA	1/9830	1/12	WELite	34:33	1:14:46	1:4
10	02:33:00	5:51	F1	Zakharova, Svetlana	Cheboksary	RUS	2/9830	2/12	WELite	36:15	1:15:52	1:4	

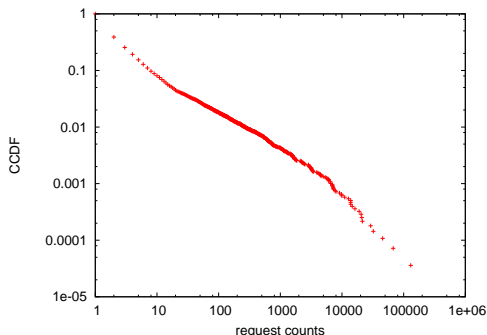
...

- ▶ Chip Time: finish time
- ▶ Category: MELite, WELite, M15-19, M20-24, ..., W15-29, W20-24, ...
 - ▶ note some runners have "No Age" for Category
- ▶ Country: 3-letter country code: e.g., JPN, USA
 - ▶ note some runners have "UK" for country-code
- ▶ check the number of the total finishers when you extract the finishers

today's exercise: CCDF plots

extract the access count of each unique content from the JAIST server access log, plot the access count distribution in CCDF

```
% ./count_contents.rb sample_access_log > contents.txt  
% ./make_ccdf.rb contents.txt > ccdf.txt
```



extracting the access count of each unique content

```
# output: URL req_count byte_count
# regular expression for apache combined log format
# host ident user time request status bytes referer agent
re = /^(S+) (\S+) (\S+) \[(.)*\] "(.*)" (\d+) (\d+|-) "(.*)" "(.*)" /
# regular expression for request: method url proto
req_re = /(\w+) (\S+) (\S+)/
contents = Hash.new([0, 0])
count = parsed = 0
ARGF.each_line do |line|
  count += 1
  if re.match(line)
    host, ident, user, time, request, status, bytes, referer, agent = $~.captures
    # ignore if the status is not success (2xx)
    next unless /2\d{2}/.match(status)
    if req_re.match(request)
      method, url, proto = $~.captures
      # ignore if the method is not GET
      next unless /GET/.match(method)
      parsed += 1
      # count contents by request and bytes
      contents[url] = [contents[url][0] + 1, contents[url][1] + bytes.to_i]
    else
      # match failed. print a warning msg
      $stderr.puts("request match failed at line #{count}: #{line.dump}")
    end
  else
    $stderr.puts("match failed at line #{count}: #{line.dump}") # match failed.
  end
end
contents.sort_by{|key, value| -value[0]}.each do |key, value|
  puts "#{key} #{value[0]} #{value[1]}"
end
$stderr.puts "# #{contents.size} unique contents in #{parsed} successful GET requests"
$stderr.puts "# parsed:#{parsed} ignored:#{count - parsed}"
```

script to convert the access count to CCDF

```
#!/usr/bin/env ruby

re = /^S+s+(\d+)\s+\d+/

n = 0
counts = Hash.new(0)
ARGF.each_line do |line|
  if re.match(line)
    counts[$1.to_i] += 1
    n += 1
  end
end

cum = 0
counts.sort.each do |key, value|
  comp = 1.0 - Float(cum) / n
  puts "#{key} #{value} #{comp}"
  cum += value
end
```

gnuplot script for plotting the content access count in CCDF

```
set logscale
set xlabel "request counts"
set ylabel "CCDF"

plot "ccdf.txt" using 1:3 notitle with points
```

summary

Class 5 Diversity and complexity (10/31)

- ▶ Long tail
- ▶ Web access and content distribution
- ▶ Power-law and complex systems
- ▶ exercise: power-law analysis

next class

Class 6 Correlation (11/7)

- ▶ Online recommendation systems
- ▶ Distance
- ▶ Correlation coefficient
- ▶ exercise: correlation analysis