# Internet Measurement and Data Analysis (4)

Kenjiro Cho

2012-10-24

## review of previous class

Class 3 Data recording and log analysis (10/17)

- Network management tools
- Data format
- Log analysis methods
- exercise: log data and regular expression

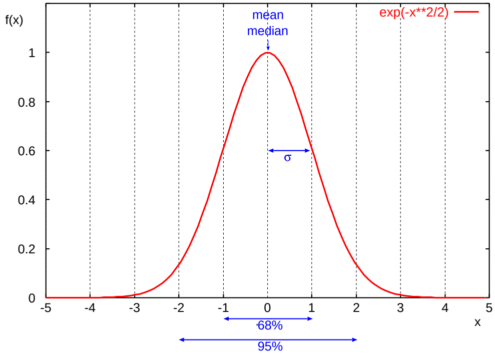
## today's topics

Class 4 Distribution and confidence intervals (10/24)

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

## normal distribution (1/2)

- also known as gaussian distribution
- defined by 2 parameters: $\mu$:mean, $\sigma^2$: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)

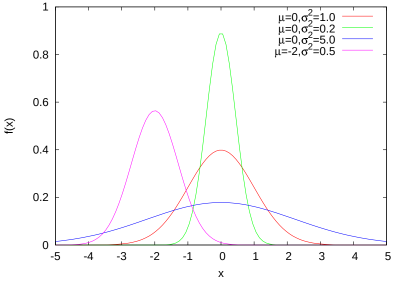
probability density function (PDF)

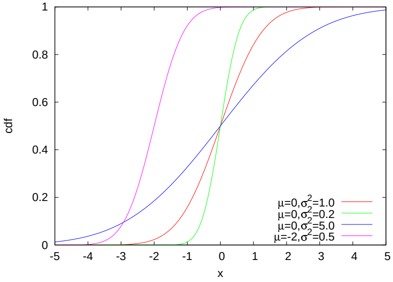
$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

cumulative distribution function (CDF)

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

$$\mu : mean, \sigma^2 : variance$$





## confidence interval

- confidence interval
  - provides probabilistic bounds
  - tells how much uncertainty in the estimate
- confidence level, significance level

$$Prob\{c_1 \leq \mu \leq c_2\} = 1 - \alpha$$

| | |
|---|---|
| $(c1, c2)$ : | *confidence interval* |
| $100(1-\alpha)$ : | *confidence level* |
| $\alpha$ : | *significance level* |

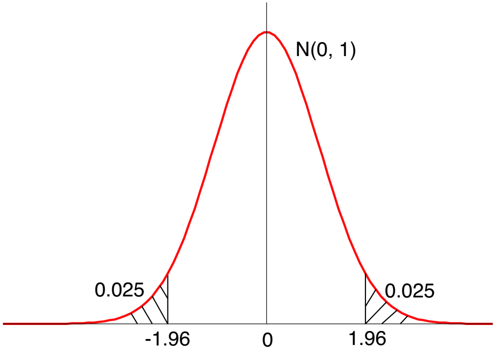
- example: with 95% confidence, the population mean is between $c1$ and $c2$
- traditionally, 95% and 99% are often used for confidence level

## 95% confidence interval

sample mean from normal distribution $N(\mu, \sigma)$ follows normal distribution $N(\mu, \sigma/\sqrt{n})$

95% confidence interval corresponds to the following area in the standard normal distribution

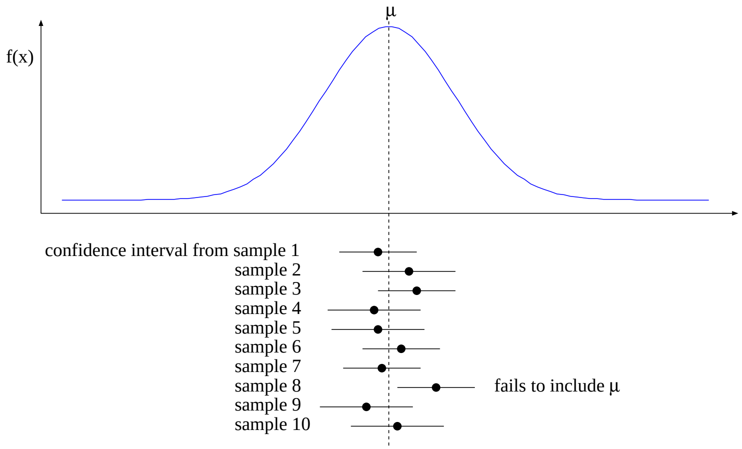
$$-1.96 \leq \frac{\bar{x} - \mu}{\sigma\sqrt{n}} \leq 1.96$$



standard normal distribution N(0, 1)

## illustration of confidence interval

- confidence level 90% means 90% samples will contain population mean in their confidence intervals



## confidence interval for mean

when sample size is large, confidence interval for population mean is

$$\bar{x} \mp z_{1-\alpha/2}\, s/\sqrt{n}$$

here, $\bar{x}$:sample mean, $s$:sample standard deviation, $n$:sample size,$\alpha$:significance level

$z_{1-\alpha/2}$:$(1-\alpha/2)$-quantile of unit normal variate

- for 95% confidence level: $z_{1-0.05/2} = 1.960$
- for 90% confidence level: $z_{1-0.10/2} = 1.645$
- example: 5 measurements of TCP throughput
  - 3.2, 3.4, 3.6, 3.6, 4.0Mbps
  - sample mean $\bar{x} = 3.56$Mbps, sample standard deviation $s = 0.30$Mbps
  - 95% confidence interval:

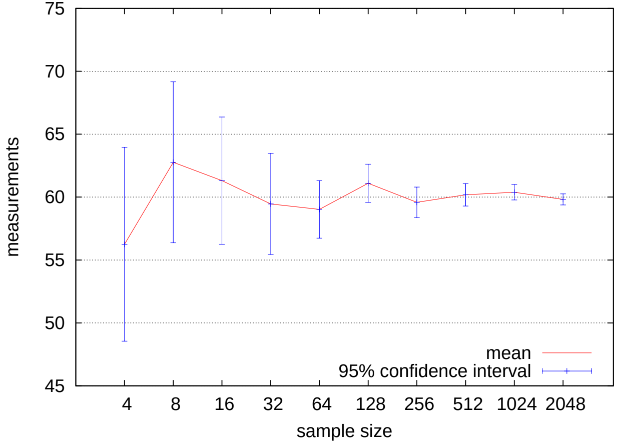
    $$\bar{x} \mp 1.96(s/\sqrt{n}) = 3.56 \mp 1.960 \times 0.30/\sqrt{5} = 3.56 \mp 0.26$$

  - 90% confidence interval:

    $$\bar{x} \mp 1.645(s/\sqrt{n}) = 3.56 \mp 1.645 \times 0.30/\sqrt{5} = 3.56 \mp 0.22$$

## confidence interval for mean and sample size

confidence interval becomes smaller as sample size increases



confidence interval with varying sample size

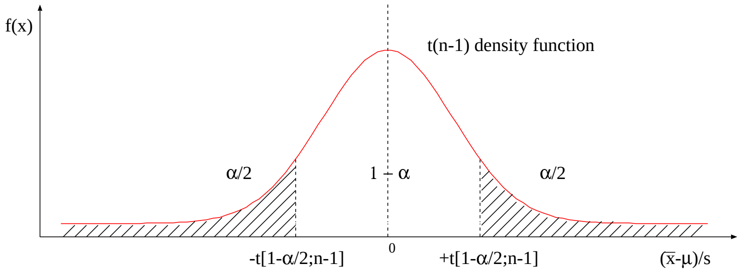
## confidence interval for mean when sample size is small

when sample size is small ($< 30$), confidence interval can be constructed only if population has normal distribution

- $(\bar{x} - \mu)/(s/\sqrt{n})$ for samples from normal population follows $t(n-1)$ distribution

$$\bar{x} \mp t_{[1-\alpha/2;n-1]}\, s/\sqrt{n}$$

here, $t_{[1-\alpha/2;n-1]}$:$(1-\alpha/2)$-quantile of a t-variate with $(n-1)$ degree of freedom



## example: confidence interval for mean when sample size is small

- example: in the previous TCP throughput measurement, confidence interval should be calculated using $t(n-1)$ distribution
  - 95% confidence interval, $n = 5$: $t_{[1-0.05/2,4]} = 2.776$

    $$\bar{x} \mp 2.776(s/\sqrt{n}) = 3.56 \mp 2.776 \times 0.30/\sqrt{5} = 3.56 \mp 0.37$$

  - 90% confidence interval, $n = 5$: $t_{[1-0.10/2,4]} = 2.132$

    $$\bar{x} \mp 2.132(s/\sqrt{n}) = 3.56 \mp 2.132 \times 0.30/\sqrt{5} = 3.56 \mp 0.29$$

## other confidence intervals

- for population variance
  - chi-square distribution with degree of freedom $(n-1)$
- for ratio of sample variances
  - F distribution with degree of freedom $(n_1 - 1, n_2 - 1)$

## how to use confidence interval

applications

- provide confidence interval to show possible range of mean
- from sample mean and stddev, compute how many trials are needed to satisfy a given confidence interval
- repeat measurement until a given confidence interval is reached

## sample size for determining mean

- how many observations $n$ is required to estimate population mean with accuracy $\pm$r% and confidence level $100(1-\alpha)\%$?
- perform preliminary test to obtain sample mean $\bar{x}$ and standard deviation $s$
- for sample size $n$, confidence interval is $\bar{x} \mp z\frac{s}{\sqrt{n}}$
- desired accuracy of $r\%$

$$\bar{x} \mp z\frac{s}{\sqrt{n}} = \bar{x}(1 \mp \frac{r}{100})$$

$$n = (\frac{100zs}{r\bar{x}})^2$$

- example: by preliminary test for TCP throughput, the sample mean is 3.56Mbps, sample standard deviation is 0.30Mbps. how many observations will be required to obtain accuracy ($<$ 0.1Mbps) with 95% confidence?

$$n = (\frac{100zs}{r\bar{x}})^2 = (\frac{100 \times 1.960 \times 0.30}{0.1/3.56 \times 100 \times 3.56})^2 = 34.6$$

## inference and hypothesis testing

the purpose of hypothesis testing

- a method to statistically test a hypothesis on population using samples

inference and hypothesis testing: both sides of the coin

- inference: predict a value to be within a range
- hypothesis testing: whether a hypothesis is accepted or rejected
  - make a hypothesis about population, compute if the hypothesis falls within the 95% confidence interval
  - accept the hypothesis if it is within the range
  - reject the hypothesis if it is outside of the range

## example: hypothesis testing

when flipping $N$ coins, we have 10 heads. In this case, can we accept a hypothesis of $N = 36$? (here, assume the distribution follows normal distribution with $\mu = N/2, \sigma = \sqrt{n}/2$)

- hypothesis: 10 heads for $N = 36$
- hypothesis testing for 95% confidence level

$$-1.96 \leq (\bar{x} - 18)/3 \leq 1.96 \qquad 12.12 \leq \bar{x} \leq 23.88$$

10 is outside of the 95% confidence interval so that the hypothesis of $N = 36$ is rejected

## discarding outliers

outliers should not be discarded blindly. investigation needed, which sometimes leads to new findings

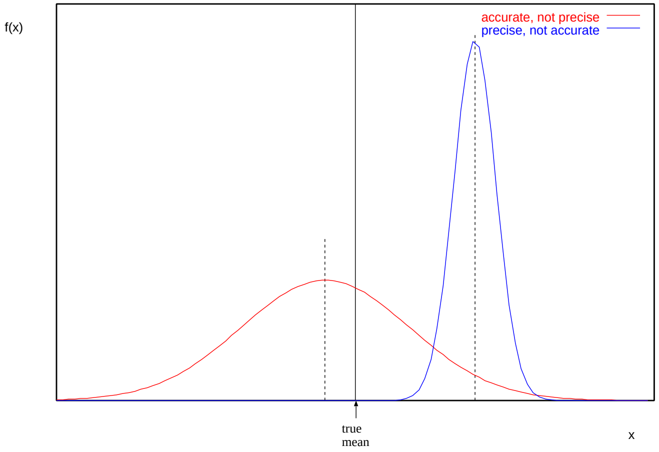
- Chauvenet's criterion: heuristic method to reject outliers
  - calculate sample mean and standard deviation from sample size $n$
  - assuming normal distribution, determine the probability $p$ of suspected data point
  - if $n \times p < 0.5$, the suspicious data point may be discarded
  - note: when $n < 50$, $s$ is not reliable. the method should not apply repeatedly
- example: 10 delay measurements: 4.6, 4.8, 4.4, 3.8, 4.5, 4.7, 5.8, 4.4, 4.5, 4.3 (sec). is it ok to discard 5.8sec?
  - $\bar{x} = 4.58, s = 0.51$
  - $t_{sus} = \frac{x_{sus} - \bar{x}}{s} = \frac{5.8 - 4.58}{0.51} = 2.4$, 2.4 times larger than $s$
  - $P(|x - \bar{x}| > 2.4s) = 1 - P(|x - \bar{x}| < 2.4s) = 1 - 0.984 = 0.016$
  - $n \times p = 10 \times 0.016 = 0.16$
  - $0.16 < 0.5$: we may discard 5.8sec

## accuracy, precision and errors

accuracy: how close to true value

precision: uncertainty in data

error: difference from true value, range of uncertainty



## various errors

measurement errors

- systematic errors (if conditions are identified, errors could be corrected)
  - instrument error, procedural error, personal bias
- random errors (noise: accuracy can be improved by repeating measurement)

calculation errors

- round-off errors (丸め誤差)
- truncation errors (打ち切り誤差)
- loss of trailing digits (情報落ち)
- cancellation of significant digits (桁落ち)
- propagation of error (誤差の伝搬)

sampling errors

- when sampling is used, true value is usually unknown
- sampling errors: errors in estimating population characteristics

## significant digits

significant digits of "1.23" is 3 ($1.225 \leq 1.23 < 1.235$)

expressions

| expressions | significant digits | |
|---|---|---|
| 12.3 | 3 | |
| 12.300 | 5 | |
| 0.0034 | 2 | |
| 1200 | 4 | (vague, $1.200 x 10^3$) |
| $2.34 x 10^4$ | 3 | |

arithmetic

- use all the available digits during calculation
  - for manual calculation, use one more digit
- apply the significant digits to the final value

basic rules

- addition/subtraction: use the smallest number of decimal places
  - $1.23 + 5.724 = 6.954 \Rightarrow 6.95$
- multiplication/division: use the smallest number of significant digits
  - $4.23 \times 0.38 = 1.6074 \Rightarrow 1.6$

## computational precision of computers

- integer (32/64bits)
  - 32bit signed integer (up to 2G)
- 32bit floating point (IEEE 754 single precision): significant digits:7
  - sign:1bit, exponent:8bits, mantissa:23bits
  - $16,000,000 + 1 = 16,000,000$!!
- 64bit floating point (IEEE 754 double precision): significant digits:15
  - sign:1bit, exponent:11bits, mantissa:52bits

## previous exercise: web access log sample data

- apache log (combined log format)
- from a JAIST server, access log for 24 hours
- about 14MB (bzip2 compressed), about 280MB after bunzip2
- 1/10 sampling
- client IP addresses are anonymized (1-to-1 mapping) for privacy
- test data (first 100 lines)

```
access log for 24 hours:
 http://www.iijlab.net/~kjc/classes/sfc2012f-measurement/sample_access_log.bz2
test data (first 100 lines):
 http://www.iijlab.net/~kjc/classes/sfc2012f-measurement/test-100lines
```

## sample data

```
143.207.214.239 - - [18/Jul/2010:23:59:53 +0900] "GET /pub/mozilla.org/firefox/releases/3.6.6/\
  update/mac/de/firefox-3.6.6.complete.mar HTTP/1.1" 206 300371 "-" "Mozilla/5.0 (Macintosh; U;\
  Intel Mac OS X 10.6; de; rv:1.9.2.3) Gecko/20100401 Firefox/3.6.3" ftp.jaist.ac.jp
161.42.4.49 - - [18/Jul/2010:23:59:20 +0900] "GET /pub/PC-BSD/8.0/i386/PCBSD8.0-x86-DVD.iso\
  HTTP/1.1" 206 58970 "http://ftp.jaist.ac.jp/pub/PC-BSD/8.0/i386" "Mozilla/4.0 (compatible;\
  MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322; .NET CLR 2.0.50727)" ftp.jaist.ac.jp
150.107.216.201 - - [18/Jul/2010:23:59:56 +0900] "GET /pub/mozilla.org/firefox/releases/3.6.6/\
  update/win32/en-GB/firefox-3.6.6.complete.mar HTTP/1.1" 206 300368 "-" "Mozilla/5.0 (Windows;\
  U; Windows NT 6.0; en-GB; rv:1.9.2.3) Gecko/20100401 Firefox/3.6.3 (.NET CLR 3.5.30729)"\
  ftp.jaist.ac.jp
22.32.128.50 - - [19/Jul/2010:00:00:00 +0900] "HEAD /project/clamav/clamav/win32/ClamAV-0.96.1\
  -64bit-beta.zip HTTP/1.0" 200 302 "http://jaist.dl.sourceforge.net/project/clamav/clamav/\
  win32/" "Wget/1.10.2 (Red Hat modified)" jaist.dl.sourceforge.net
137.29.144.83 - - [19/Jul/2010:00:00:00 +0900] "GET /pub/mozilla.org/thunderbird/releases/\
  2.0.0.24/update/win32/en-US/thunderbird-2.0.0.24.complete.mar HTTP/1.1" 200 65845 "-"\
  "Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.22) Gecko/20090605 Thunderbird/\
  2.0.0.22" ftp.jaist.ac.jp
22.32.128.50 - - [19/Jul/2010:00:00:00 +0900] "HEAD /project/clamav/clamav/win32/Clamunrar-\
  0.96.zip HTTP/1.0" 200 298 "http://jaist.dl.sourceforge.net/project/clamav/clamav/win32/"\
  "Wget/1.10.2 (Red Hat modified)" jaist.dl.sourceforge.net
209.235.74.175 - - [18/Jul/2010:23:59:52 +0900] "GET /pub/mozilla.org/firefox/releases/3.6.6/\
  update/win32/en-US/firefox-3.6.6.complete.mar HTTP/1.1" 206 300368 "-" "Mozilla/5.0 (Windows;\
  U; Windows NT 6.1; en-US; rv:1.9.2) Gecko/20100115 Firefox/3.6" ftp.jaist.ac.jp
153.42.115.45 - - [18/Jul/2010:23:59:56 +0900] "GET /pub/mozilla.org/firefox/releases/3.5.10/\
  update/win32/pl/firefox-3.5.10.complete.mar HTTP/1.1" 206 300368 "-" "Mozilla/5.0 (Windows;\
  U; Windows NT 6.0; pl; rv:1.9.1.5) Gecko/20091102 Firefox/3.5.5 (.NET CLR 3.5.30729)"\
  ftp.jaist.ac.jp
...
```

## previous exercise: plotting request counts over time

- use the sample data
- extract request counts and transferred bytes with 5 minutes bins
- plot the results

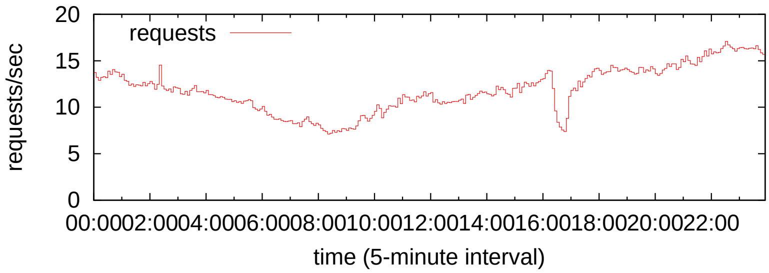
```
% ruby parse_accesslog.rb sample_access_log > access-5min.txt
% more access-5min.txt
2010-07-18T16:55 1 600572285
...
2010-07-18T23:55 463 2128020418
2010-07-19T00:00 4123 1766135158
2010-07-19T00:05 3963 1857342919
2010-07-19T00:10 3871 2171231118
2010-07-19T00:15 3965 4378143224
...
% gnuplot
gnuplot> load 'access.plt'
```

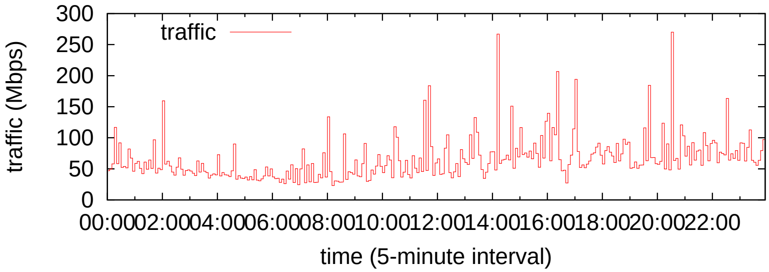
## extract request counts and transferred bytes with 5 minutes bins

```
#!/usr/bin/env ruby
require 'date'

# regular expression for apache common log format
#   host ident user time request status bytes
re = /^(\S+) (\S+) (\S+) \[(.*?)\] "(.*?)" (\d+) (\d+|-)/
timebins = 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 = $~.captures
    # ignore if the status is not success (2xx)
    next unless /2\d{2}/.match(status)
    parsed += 1
    # parse timestamp
    ts = DateTime.strptime(time, '%d/%b/%Y:%H:%M:%S %z')
    # create the corresponding key for 5-minutes timebins
    rounded = sprintf("%02d", ts.min.to_i / 5 * 5)
    key = ts.strftime("%Y-%m-%dT%H:#{rounded}")
    # count by request and byte
    timebins[key] = [timebins[key][0] + 1, timebins[key][1] + bytes.to_i]
  else
    # match failed
    $stderr.puts("match failed at line #{count}: #{line.dump}")
  end
end
timebins.sort.each do |key, value|
  puts "#{key} #{value[0]} #{value[1]}"
end
$stderr.puts "parsed:#{parsed} ignored:#{count - parsed}"
```

## plot graphs of request counts and transferred bytes





## gnuplot script

- put 2 graphs together using multiplot

```
set xlabel "time (5-minute interval)"
set xdata time
set format x "%H:%M"
set timefmt "%Y-%m-%dT%H:%M"
set xrange ['2010-07-19T00:00':'2010-07-19T23:55']
set key left top

set multiplot layout 2,1

set yrange [0:20]
set ylabel "requests/sec"
plot "access-5min.txt" using 1:($2/300) title 'requests' with steps

set yrange [0:300]
set ylabel "traffic (Mbps)"
plot "access-5min.txt" using 1:($3*8/300/1000000) title 'traffic' with steps

unset multiplot
```

## 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 u and standard deviation s, 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 "$\pm 1.960 \frac{s}{\sqrt{n}}$".
    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 onterval 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

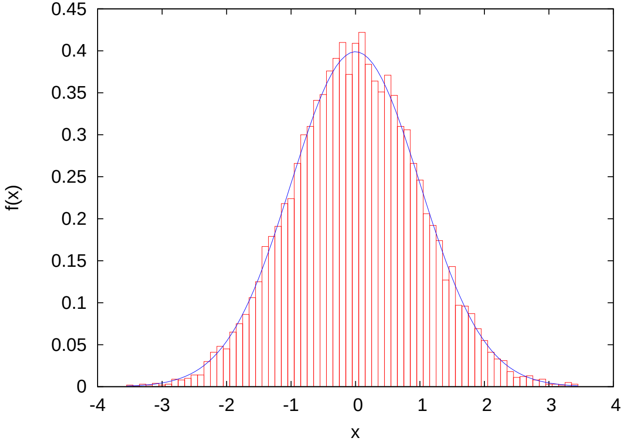
```
# usage: box-muller.rb [n [m [s]]]
n = 1 # number of samples to output
mean = 0.0
stddev = 1.0

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

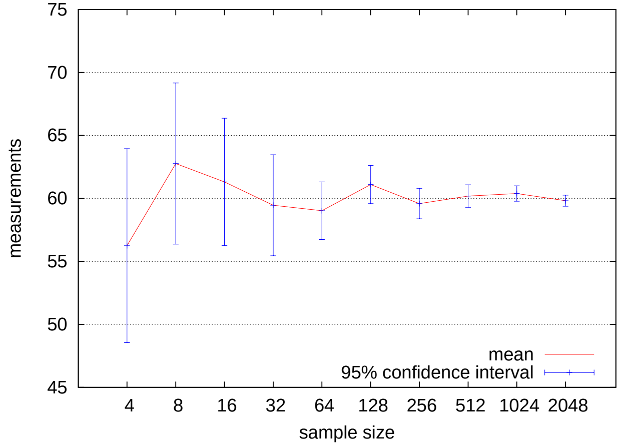
```
set boxwidth 0.1
set xlabel "x"
set ylabel "f(x)"
plot "box-muller-histogram.txt" using 1:($2/1000) with boxes notitle, \
     1/sqrt(2*pi)*exp(-x**2/2) notitle with lines linetype 3
```

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

```
      Chip    Pace                                               Gender   Category       @10km  @21.1  @3
Place Time    /mi      #    Name              City         ST CNT Plce/Tot Plc/Tot Category Split1 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

## summary

Class 4 Distribution and confidence intervals

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

## next class

Class 5 Diversity and complexity (10/31)

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