インターネット計測とデータ解析 第4回

長 健二朗

2012年4月27日

前回のおさらい

データの収集と記録

- ネットワーク管理ツール
- データフォーマット
- ログ解析手法
- 演習: ログデータと正規表現

今日のテーマ

分布と信頼区間

- サンプリング
- 正規分布
- 信頼区間と検定
- 分布の生成
- 演習: 分布の生成、信頼区間
- **課題 1**

サンプリング

- 全数調査: ほとんどの場合は非現実的
- サンプリングが必要になる

インターネット計測におけるサンプリング

- 測定場所
- 時間、期間
- パケット、フロー

パケットのサンプリング方法

- カウンタベースの 1/N サンプリング (決定論的)
  - 実装が簡単、広く使われている
  - 測定対象と同期してしまう可能性
- 確率的 1/N サンプリング
  - パケットごとにサイコロを振って決める
- 時間によるサンプリング
  - 例: 毎時最初の 1 分を計測
- フローベースのサンプリング
  - 新しいフローは確率的にサンプル
  - 選んだフローのパケットは全部測定
  - フローの挙動解析が可能
- 他にも様々な方法が存在

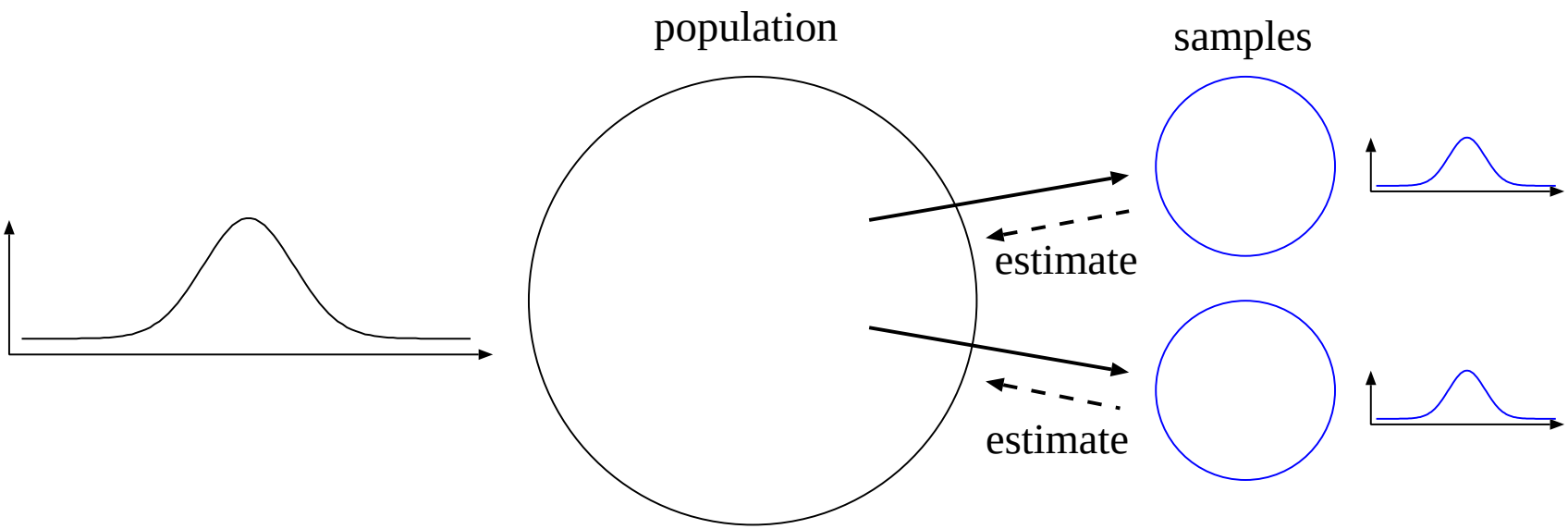
サンプリング: 標本と母集団

要約と推測

- 要約統計量 (平均、標準偏差など) は分布の特徴を要約して表す数値
- 推測統計は標本 (サンプル) から母集団の性質を統計的に推測する

母集団 (population): 全体のデータ、多くの場合入手不可能

- 標本 (sample) から母集団の性質を推定する必要
- 変数: 母集団の特徴 (固定)
- 統計: 標本からの推定値 (ゆらぎを持つ変数)



期待値

確率変数 X の期待値 $E(X)$ (平均を表す)

- 離散型

$$E(X) = \mu = \sum_{i=1}^{n} x_i p_i$$

- 連続型

$$E(X) = \mu = \int_{-\infty}^{\infty} x f(x) dx$$

期待値の性質

- $E(c) = c$
- $E(X + c) = E(X) + c$
- $E(cX) = cE(X)$
- $E(X + Y) = E(X) + E(Y)$

標本平均

- 標本平均 (sample mean): $\bar{x}$

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

- 標本分散 (sample variance): s^2

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

- 標本標準偏差 (sample standard deviation): s
- 注: 二乗和を n ではなく $(n-1)$ で割る
  - 自由度 (degree of freedom): 二乗和の独立変数は $\bar{x}$ があるため 1 減る

大数の法則と中心極限定理

大数の法則

- サンプル数が増えるに従い標本平均は母平均に近付く

中心極限定理

- 元の分布に関わらず (十分なサンプル数があれば) 標本平均は近似的に正規分布に従う $N(\mu, \sigma^2/n)$
- 母集団が正規分布の場合は、n が小さくてもこの関係が成立する

標準誤差 (standard error)

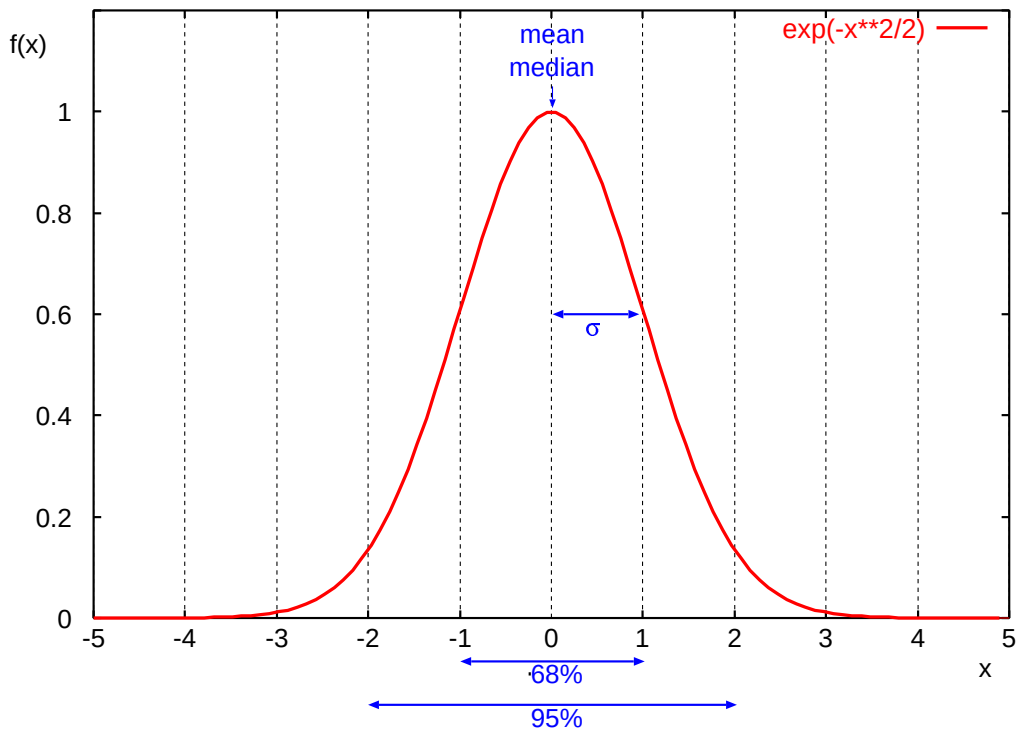
標準誤差: 標本平均の標準偏差 (SE)

$$SE = \sigma/\sqrt{n}$$

- サンプル数 n を増やすと精度が改善
  - 標準誤差は $1/\sqrt{n}$ に (ゆっくり) 減少
- 正規母集団 $N(\mu, \sigma)$ から取った標本平均の分布は平均 μ 標準偏差 $\mathrm{SE} = \sigma/\sqrt{n}$ の正規分布となる

正規分布 (normal distribution) 1/2

- つりがね型の分布、ガウス分布とも呼ばれる
- 2 つの変数で定義: 平均 μ、分散 σ^2
- 乱数の和は正規分布に従う
- 標準正規分布: $\mu = 0, \sigma = 1$
- 正規分布ではデータの
  - 68%は $(mean \pm stddev)$
  - 95%は $(mean \pm 2stddev)$ の範囲に入る



正規分布 (normal distribution) 2/2

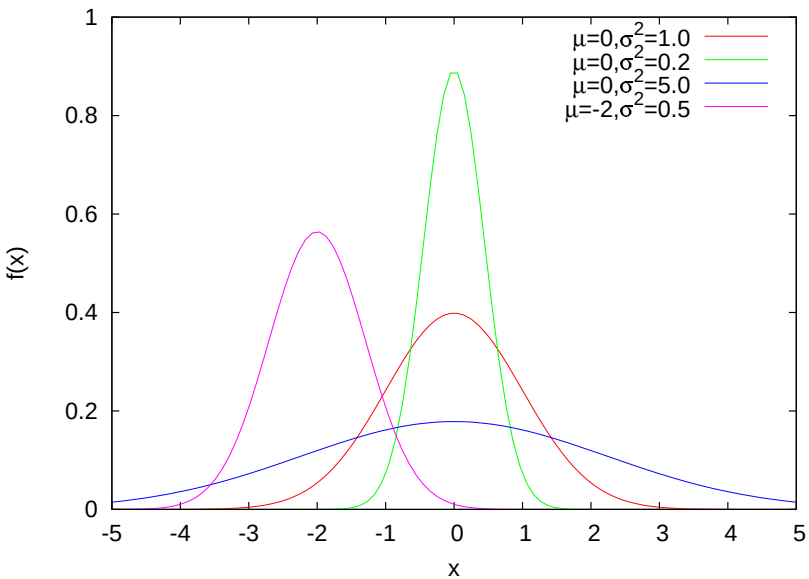
確率密度関数 (PDF)

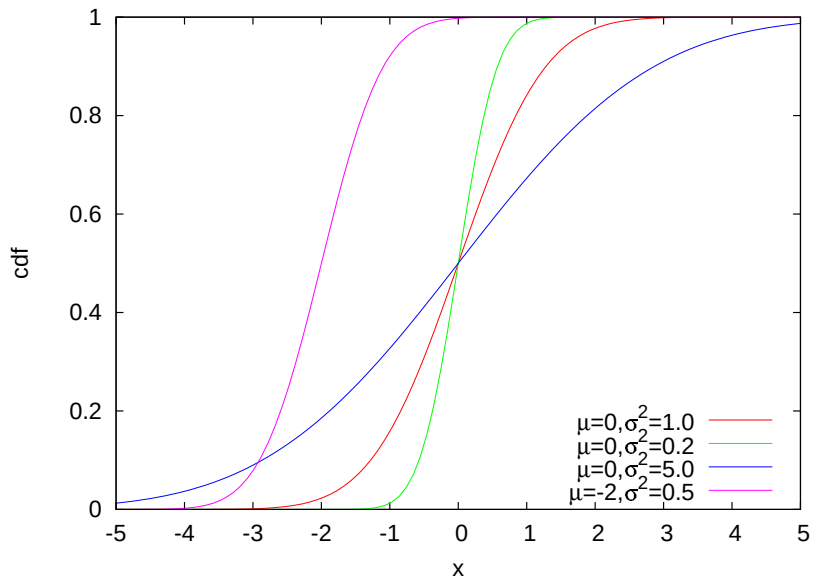
$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2}$$

累積分布関数 (CDF)

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

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





信頼区間 (confidence interval)

- 信頼区間 (confidence interval)
  - 統計的に真値に範囲を示す
  - 推定値の確かさ、不確かさを示す
- 信頼度 (confidence level) 有意水準 (significance level)

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

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

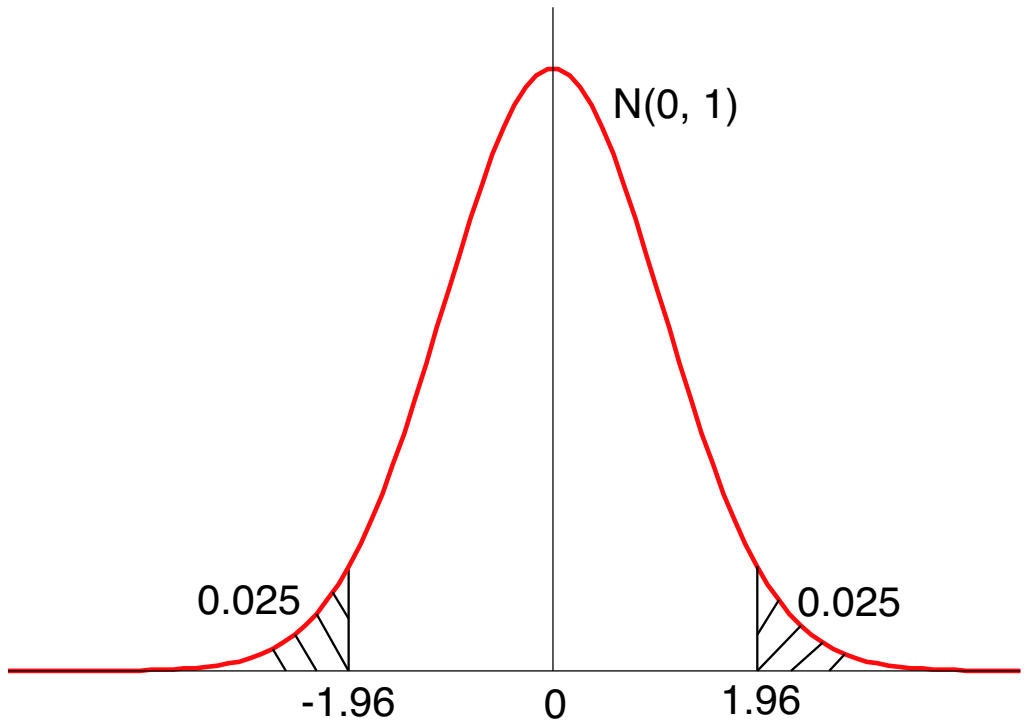
- 例: 信頼度 95% で、母平均は、$c1$ と $c2$ の間に存在
- 慣習として、信頼度 95% と 99% がよく使われる

95%信頼区間

正規母集団 $N(\mu, \sigma)$ から得られた標本平均 $\bar{x}$ は正規分布 $N(\mu, \sigma/\sqrt{n})$ に従う

95%信頼区間は標準正規分布の以下の部分を意味する

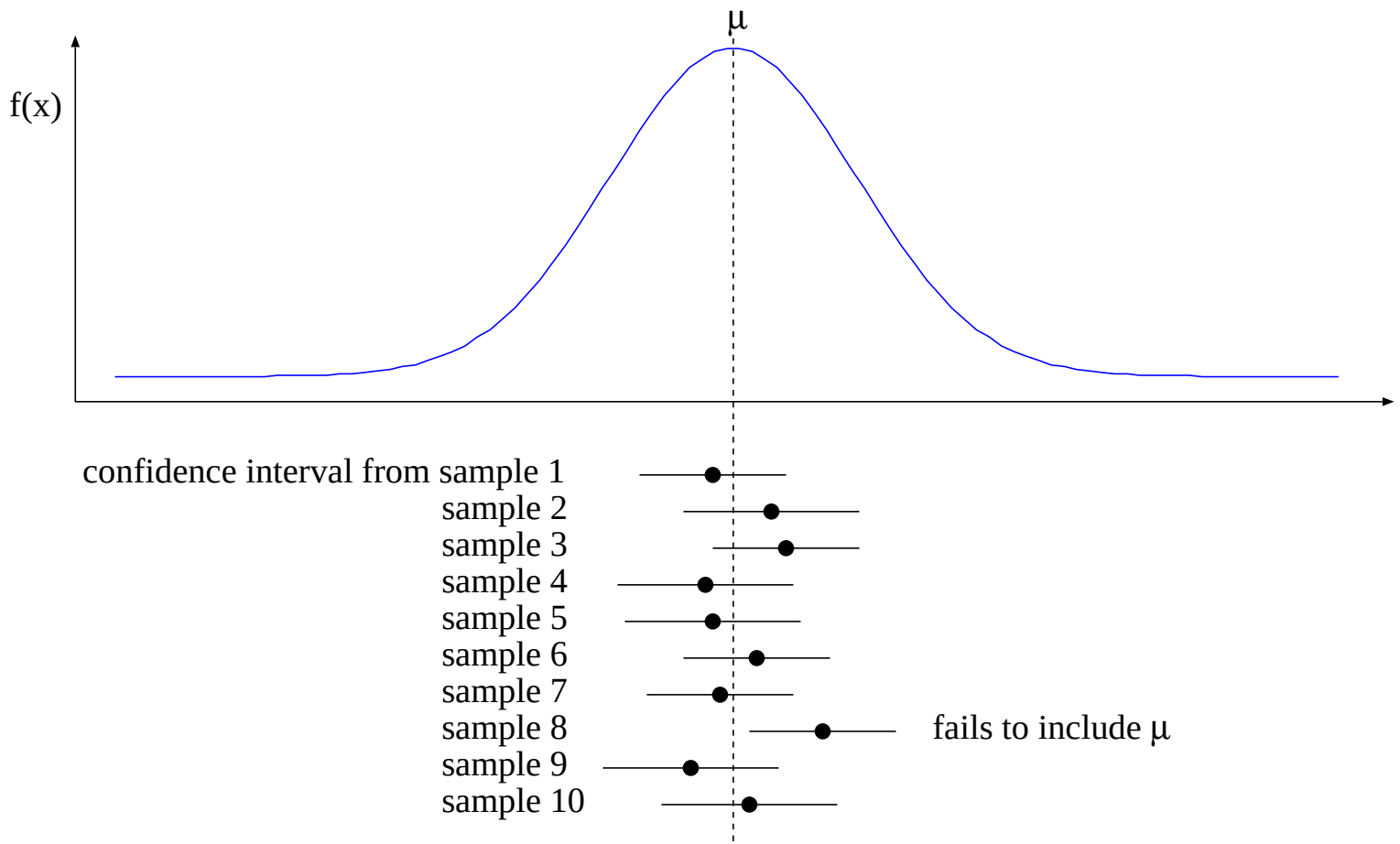
$$-1.96 \leq \frac{\bar{x} - \mu}{\sigma\sqrt{n}} \leq 1.96$$



標準正規分布 N(0, 1)

信頼区間の意味

- 信頼度 90% とは、90% の確率で母平均が信頼区間内に存在すること



平均値の信頼区間

サンプルサイズが大きければ、母平均の信頼区間は、

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

ここで、$\bar{x}$:標本平均 s:標本標準偏差 n:標本数 α:有意水準
$z_{1-\alpha/2}$:標準正規分布における $(1-\alpha/2)$ 領域の境界値

- 信頼度 95% の場合: $z_{1-0.05/2} = 1.960$
- 信頼度 90% の場合: $z_{1-0.10/2} = 1.645$
- 例: TCP スループットを 5 回計測
  - 3.2, 3.4, 3.6, 3.6, 4.0Mbps
  - 標本平均:$\bar{x} = 3.56$Mbps 標本標準偏差:$s = 0.30$Mbps
  - 95%信頼区間:

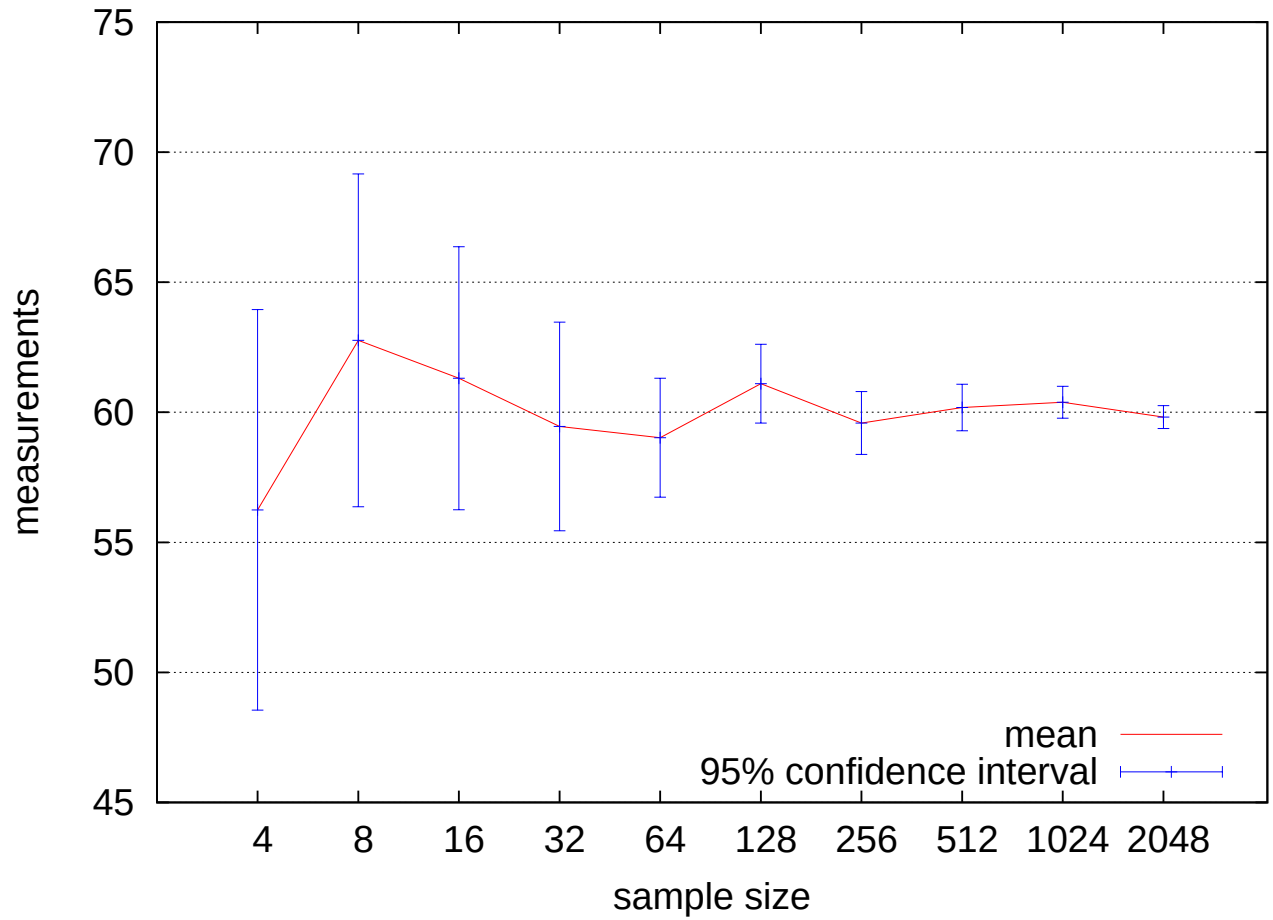
$$\bar{x} \mp 1.96(s/\sqrt{n}) = 3.56 \mp 1.960 \times 0.30/\sqrt{5} = 3.56 \mp 0.26$$

  - 90%信頼区間:

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

平均値の信頼区間とサンプル数

サンプル数が増えるに従い、信頼区間は狭くなる



平均値の信頼区間のサンプル数による変化

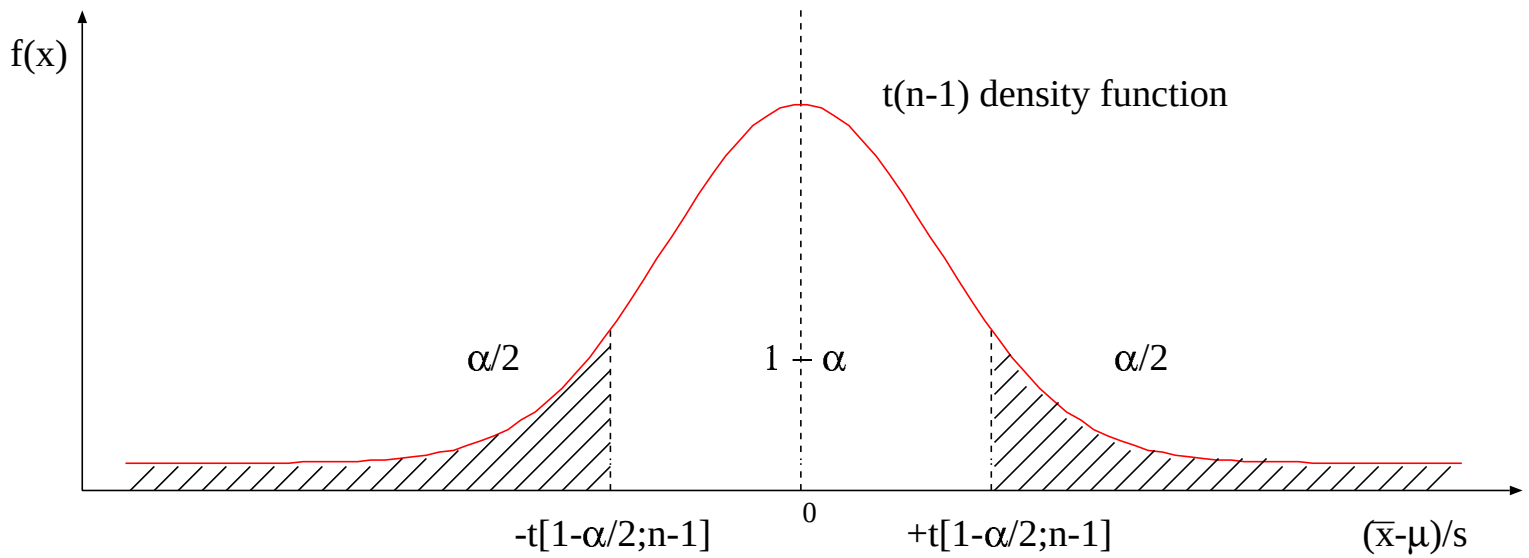
サンプル数が少ない場合の平均値の信頼区間

サンプル数が少ない (< 30) 場合、母集団が正規分布に従う場合に限って、信頼区間を求める事ができる

- 正規分布からサンプルを取った場合、標準誤差 $(\bar{x} - \mu)/(s/\sqrt{n})$ は $t(n-1)$ 分布となる

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

ここで、$t_{[1-\alpha/2;n-1]}$ は 自由度 $(n-1)$ の t 分布における $(1-\alpha/2)$ 領域の境界値



サンプル数が少ない場合の平均値の信頼区間の例

- 例: 前述の TCP スループット計測では、$t(n-1)$ 分布を使った信頼区間の計算をする必要
  - 95%信頼区間 $n = 5$: $t_{[1-0.05/2,4]} = 2.776$

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

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

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

他の信頼区間

- 母分散:
  - 自由度 $(n-1)$ の χ^2 分布
- 標本分散の比:
  - 自由度 (n_1-1, n_2-1) の F 分布

信頼区間の応用

応用例

- 平均値の推定範囲を示す
- 平均と標準偏差から、必要な信頼区間を満足するために何回試行が必要か求める
- 必要な信頼区間を満足するまで計測を繰り返す

平均を得るために必要なサンプル数

- 信頼度 $100(1-\alpha)$ で $\pm r\%$ の精度で母平均を推定するためには何回の試行 n が必要か？
- 予備実験を行い 標本平均 $\bar{x}$ と 標準偏差 s を得る
- サンプルサイズ n、信頼区間 $\bar{x} \mp z\frac{s}{\sqrt{n}}$、必要な精度 $r\%$

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

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

- 例: TCP スループットの予備計測で、標本平均 3.56Mbps、標本標準偏差 0.30Mbps を得た。
  信頼度 95%、精度 (< 0.1Mbps) で平均を得るためには何回測定する必要があるか？

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

推定と仮説検定

仮説検定 (hypothesis testing) の目的

- 母集団について仮定された命題を標本に基づいて検証

推定と仮説検定は裏表の関係

- 推定: ある範囲に入ることを予想
- 仮説検定: 仮説が採用されるか棄却されるか
  - 母集団に入るという仮説を立て、その仮説が 95%信頼区間に入るかを計算
  - 区間内であれば仮説は採用される
  - 区間外では仮説は棄却される

検定の例

N 枚のコインを投げて表が 10 枚でた。 この場合の N として 36 枚はあり得るか？ (ただし分布は $\mu = N/2, \sigma = \sqrt{n}/2$ の正規分布にしたがうものとする)

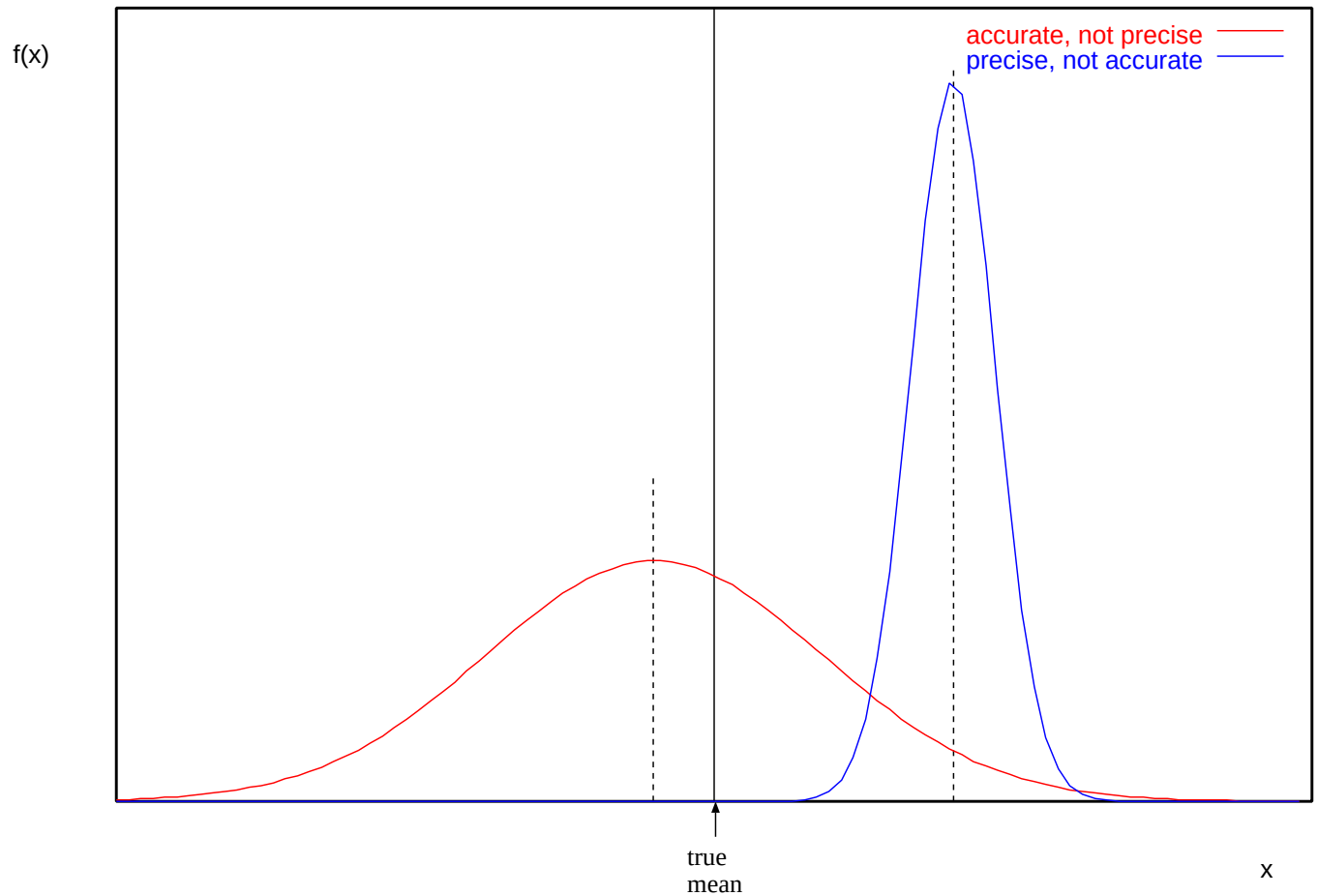
- 仮説: $N = 36$ で表が 10 枚出る
- 95%信頼度で検定

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

10 は 95%区間の外側にあるので 95%信頼度では $N = 36$ という仮説は棄却される

正確度と精度、誤差

正確度 (accuracy): 測定値と真値とのずれ
精度 (precision): 測定値のばらつきの幅
誤差 (error): 真値からのずれ、その不確かさの範囲



いろいろな誤差

測定誤差

- 系統誤差 (条件を把握できれば補正可能)
  - 器械的誤差、理論的誤差、個人的誤差
- 偶然誤差 (ノイズ、観測を繰り返せば精度向上)

計算誤差

- まるめ誤差
- 打ち切り誤差
- 情報落ち
- 桁落ち
- 誤差の伝搬

サンプリング誤差

- 標本調査を行う場合、普通は真値は不明
- 標本誤差: 真値との差の確率的なばらつきの幅

有効数字と有効桁数

1.23 の有効数字は 3 桁 ($1.225 \leq 1.23 < 1.235$)

表記

表記	有効桁数	
12.3	3	
12.300	5	
0.0034	2	
1200	4	(あいまい、$1.200x10^3$)
$2.34x10^4$	3	

計算

- 計算途中は桁数が大きいまま計算
  - 筆算などの場合は 1 桁多く取ればよい
- 最終的な数字に有効桁数を適用

基本ルール

- 加減算: 桁数が少ないものに合わせる
  - $1.23 + 5.724 = 6.954 \Rightarrow 6.95$
- 乗除算: もとの有効数字が最も少ないものに合わせる
  - $4.23 \times 0.38 = 1.6074 \Rightarrow 1.6$

コンピュータの計算精度

- integer (32/64bits)
  - 32bit signed integer (2G までしかカウントできない)
- 32bit floating point (IEEE 754 single precision): 有効桁数 7
  - sign:1bit, exponent:8bits, mantissa:23bits
  - $16,000,000 + 1 = 16,000,000$!!
- 64bit floating point (IEEE 754 double precision): 有効桁数 15
  - sign:1bit, exponent:11bits, mantissa:52bits

前回の演習: Web アクセスログ サンプルデータ

- apache log (combined log format)
- 自称日本最強のミラーサーバ
- ソフトウェア配布が主なので普通の web server ではない
- ftp という名前だが、http がメイン
- 約 14MB(bzip2 圧縮)、 解凍後は約 280MB
- 1/10 sampling
- クライアント IP アドレスはプライバシーを考慮して匿名化 (1-to-1 mapping)

```
access log for 24 hours:
 http://www.iijlab.net/~kjc/classes/sfc2012s-measurement/sample_access_log.bz2
test data (first 100 lines):
 http://www.iijlab.net/~kjc/classes/sfc2012s-measurement/test-100lines
```

前回の演習: サンプルアクセスログ

```
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
...
```

前回の演習: リクエスト数の時系列プロット

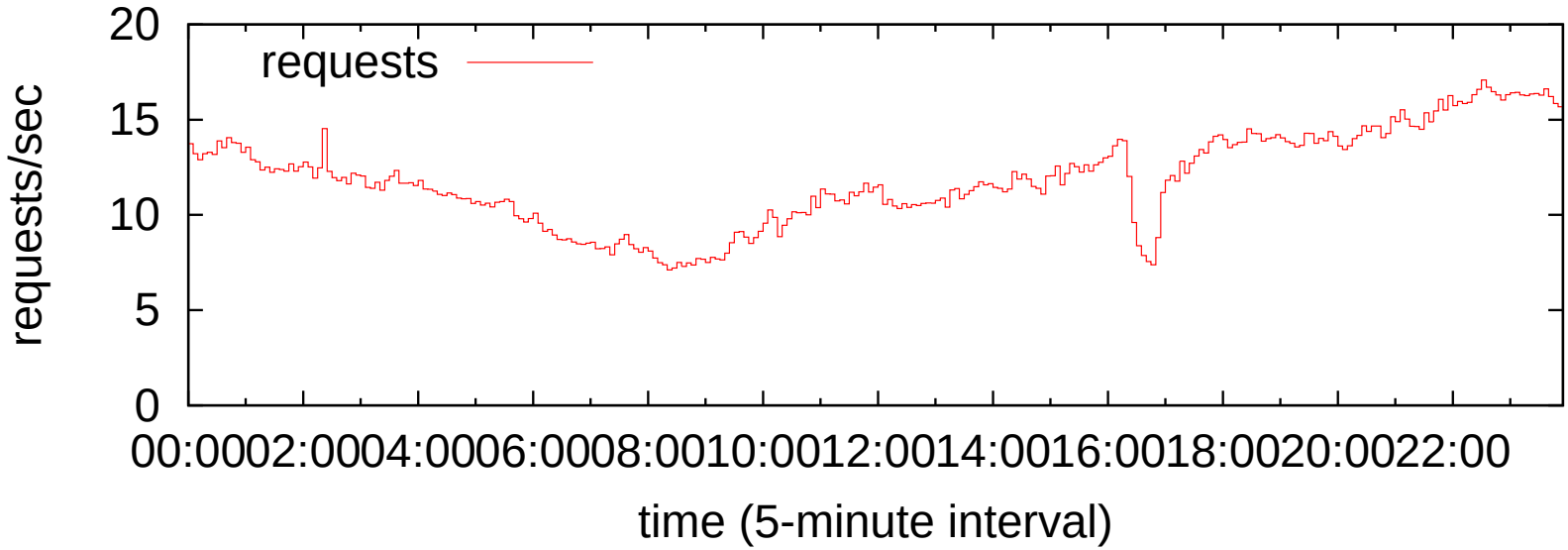
- サンプル Web アクセスログを使う
- リクエスト数と転送バイト数を 5 分間隔で抽出
- 結果をプロット

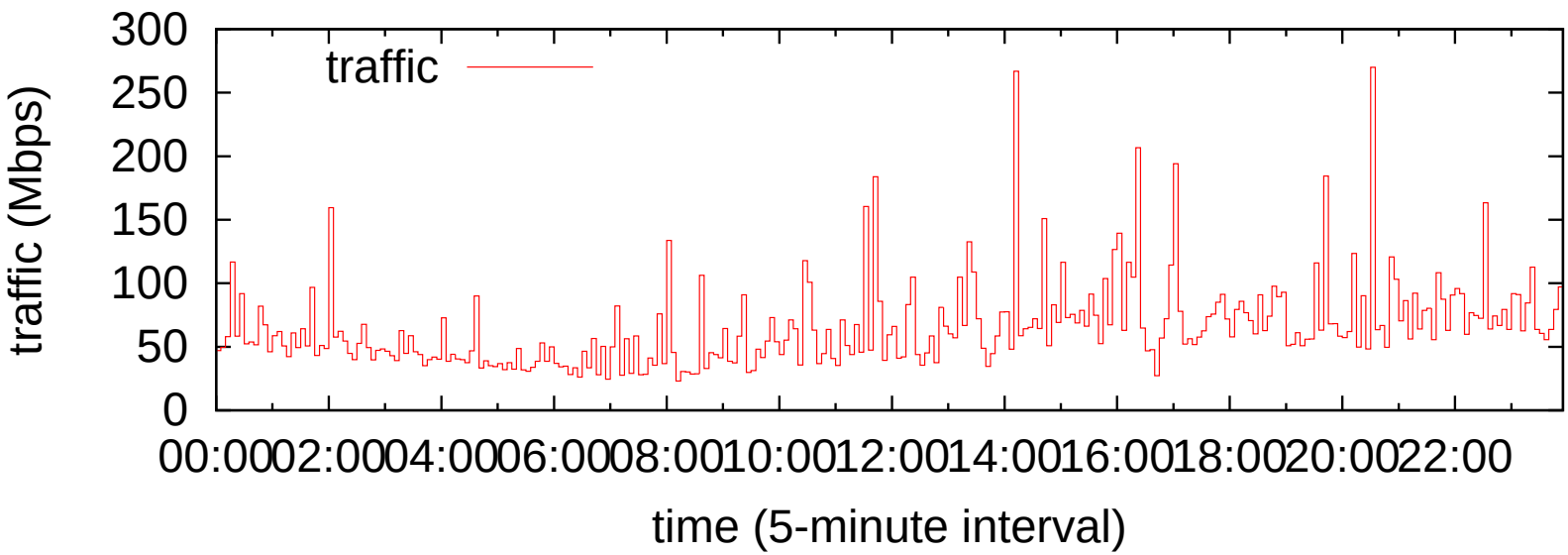
```
% ruby parse_accesslog.rb sample_access_log > access-5min.txt
% more access-5min.txt
2010-07-18T16:55 1 600572285
...
2010-07-18T23:55 463 2128020418
2010-07-19T00:00 4123 1766135158
2010-07-19T00:05 3963 1857342919
2010-07-19T00:10 3871 2171231118
2010-07-19T00:15 3965 4378143224
...
% gnuplot
gnuplot>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

前回の演習: 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}"
```

前回の演習: plot graphs of request counts and transferred bytes





前回の演習: gnuplot script

- multiplot 機能で 2 つのプロットをまとめる

```
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
```

演習: 正規乱数の生成

- 正規分布に従う疑似乱数の生成
  - 一様分布の疑似乱数生成関数 (ruby の rand など) を使って、平均 u、標準偏差 s を持つ疑似乱数生成プログラムを作成
- ヒストグラムの作成
  - 標準正規分布に従う疑似乱数を生成し、そのヒストグラム作成、標準正規分布であることを確認する
- 信頼区間の計算
  - サンプル数によって信頼区間が変化することを確認
    疑似正規乱数生成プログラムを用いて、平均 60, 標準偏差 10 の正規分布に従う乱数列を 10 種類作る。サンプル数 n = 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048 の乱数列を作る。
  - 標本から母平均の区間推定
    この 10 種類の乱数列のそれぞれから、母平均の区間推定を行え。信頼度 95%で、信頼区間 "± 1.960 s/sqrt(n)" を用いよ。10 種類の結果をひとつの図にプロットせよ。 X 軸にサンプル数を Y 軸に平均値をとり、それぞれのサンプルから推定した平均とその信頼区間を示せ

box-muller 法による正規乱数生成

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: 三角関数を使わない近似
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)

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

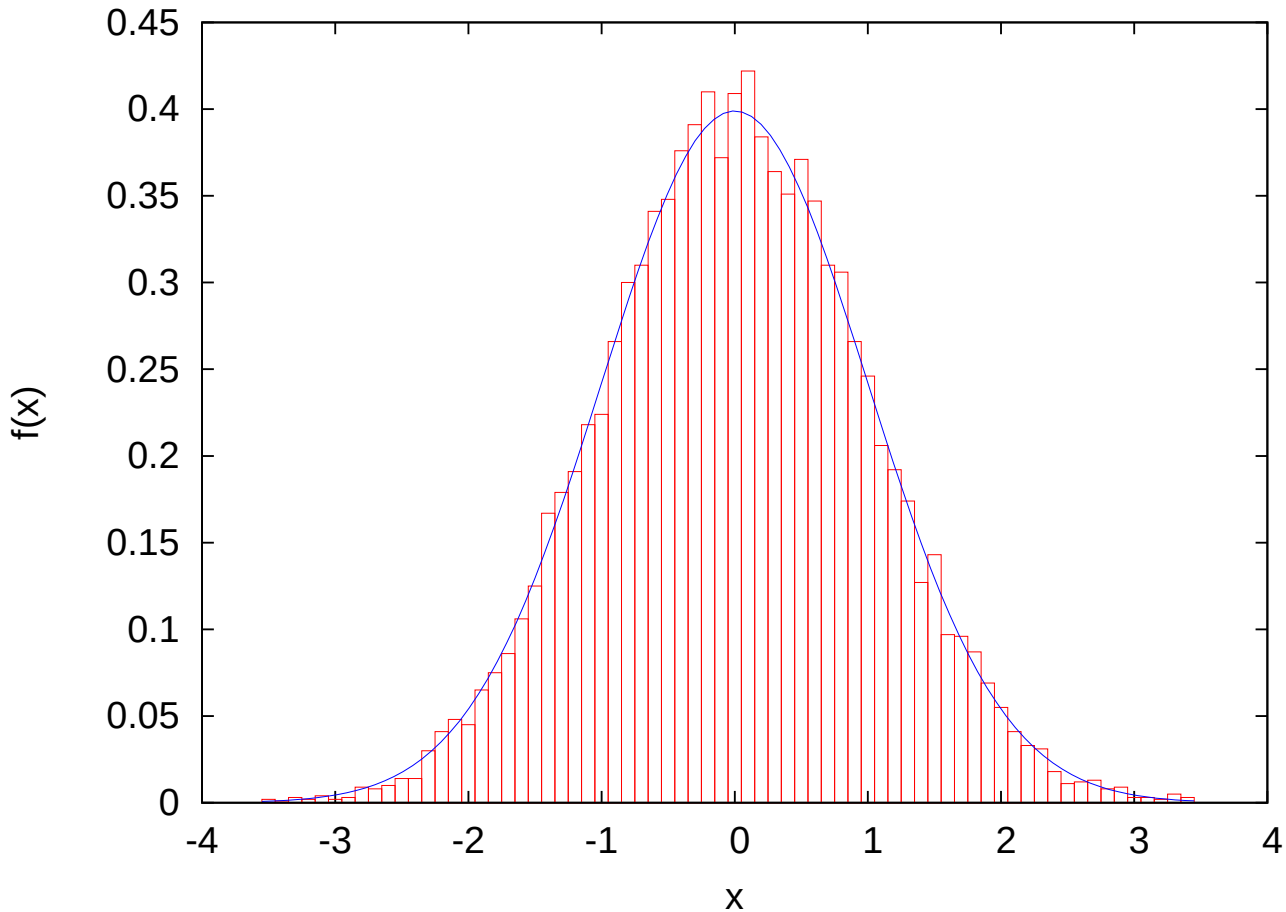
$$z_1 = u_1\sqrt{\frac{-2\ln s}{s}}$$

box-muller 法による正規乱数生成コード

```
# 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
```

正規乱数のヒストグラム作成

- 標準正規乱数のヒストグラムを作成し、正規分布であることを確認する
- 標準正規乱数を 10,000 個生成し、小数点 1 桁のビンでヒストグラムを作成



ヒストグラムの作成

- 少数点以下１桁でヒストグラムを作成する

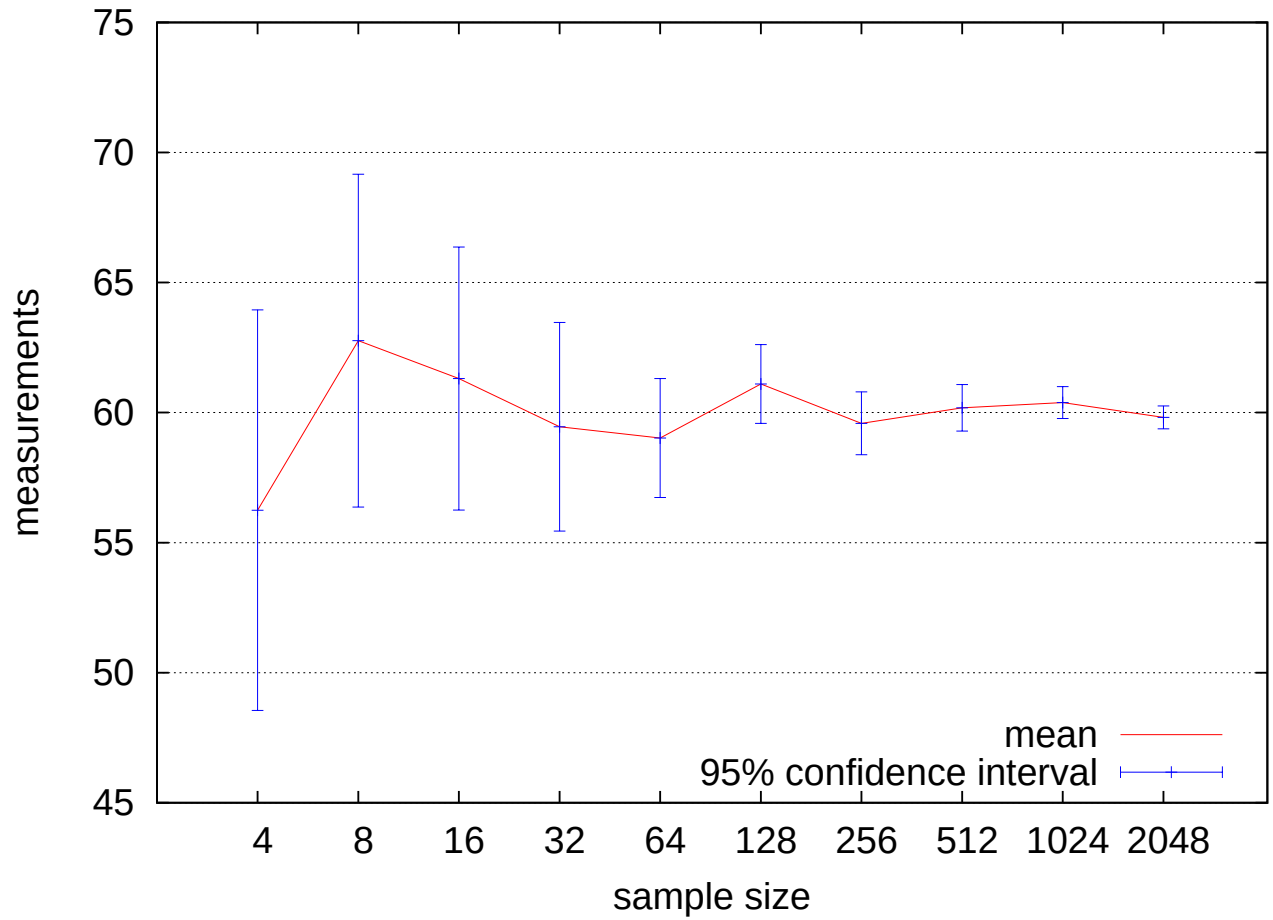
```
#
# create histogram: bins with 1 digit after the decimal point
#

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
```

正規乱数のヒストグラムのプロット

```
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
```

平均値の信頼区間とサンプル数の検証

サンプル数が増えるに従い、信頼区間は狭くなる



平均値の信頼区間のサンプル数による変化

課題 1: ホノルルマラソン完走時間のプロット

- ねらい: 実データから分布を調べる
- データ: 2011 年のホノルルマラソンの記録
  - http://results.sportstats.ca/res2011/honolulu.htm
  - 完走者 19,104 人
- 提出項目
  1. 全完走者、男性完走者、女性完走者それぞれの、完走時間の平均、標準偏差、中間値
  2. それぞれの完走時間のヒストグラム
     - 3 つのヒストグラムを別々の図に書く
     - ビン幅は 10 分にする
     - 3 つのプロットは比較できるように目盛を合わせること
  3. それぞれの CDF プロット
     - ひとつの図に 3 つのプロットを書く
  4. オプション
     - 年代別や国別の CDF プロットなど自由
  5. 考察
     - データから読みとれることを記述
- 提出形式: レポートをひとつの PDF ファイルにして SFC-SFS から提出
- 提出〆切: 2012 年 5 月 14 日

ホノルルマラソンデータ

データフォーマット

```
      Chip    Pace                                                Gender   Category       @10km  @21.1  @3
Place Time    /mi     #    Name                City          ST CNT Plce/Tot Plc/Tot Category Split1 Split2 S
----- -------- ---- ---- -------------------- ------------ -- --- -------- ------- ------ ----- ------- ---
    1 02:14:55 5:09    1 Chelimo, Nicholas    Ngong Hills     KEN  1/10191  1/11   MElite 31:25 1:07:46 1:3
    2 02:14:58 5:10    4 Ivuti, Patrick       Kangundo        KEN  2/10191  2/11   MElite 31:25 1:07:47 1:3
    3 02:15:40 5:11   11 Boit, Josphat        Fayetteville AR USA  3/10191  3/11   MElite 31:24 1:07:46 1:3
    4 02:18:12 5:17    9 Kimutai, Kiplimo     Eldoret         KEN  4/10191  4/11   MElite 31:24 1:07:46 1:3
    5 02:19:21 5:20    5 Kiptoo Kolum, B      Kapsabet        KEN  5/10191  5/11   MElite 31:24 1:07:46 1:3
    6 02:24:40 5:32    2 Mundi, Jimmy         Kangundo        KEN  6/10191  6/11   MElite 31:25 1:07:49 1:3
    7 02:31:41 5:48  104 Girma, Woynishet     Addis Ababa     ETH  1/9116   1/14   WElite 35:21 1:16:16 1:4
    8 02:31:43 5:48 8189 Puzey, Thomas        Laie         HI USA  7/10191  1/1044 M25-29 35:20 1:16:14 1:4
    9 02:31:53 5:48  106 Mekonnindemissie, M  Albuqurque  NM USA  2/9116   2/14   WElite 35:20 1:16:16 1:4
   10 02:31:55 5:48  110 Galimova, Valentina  Perm            RUS  3/9116   3/14   WElite 35:21 1:16:15 1:4
...
```

- Chip Time: 完走時間
- Category: MElite, WElite, M15-19, M20-24, ..., W15-29, W20-24, ...
  - "No Age"となっている人がいるので注意
- Country: 3-letter country code: e.g., JPN, USA
  - "UK"が交じっているので注意
- 完走者を抽出したら、総数が合っているかチェックすること

まとめ

分布と信頼区間

- サンプリング
- 正規分布
- 信頼区間と検定
- 分布の生成
- 演習: 分布の生成、信頼区間
- **課題 1**

次回予定

第 5 回 多様性と複雑さ (5/11)

- ロングテール
- Web アクセスとコンテンツ分布
- べき乗則と複雑系
- 演習: べき乗則解析