

インターネット計測とデータ解析 第2回

長 健二郎

2013年4月17日

前回のおさらい

本授業のテーマ

- ▶ いろいろな切口からインターネットとデータ解析を考える
 - ▶ 容易に計測できないものをどう計るか
 - ▶ 大量データからいかに情報を抽出する

第1回 イントロダクション (4/10)

- ▶ ビッグデータと集合知
- ▶ インターネット計測
- ▶ 大規模データ解析
- ▶ 演習: ruby 入門

今日のテーマ

第2回 データとばらつき

- ▶ 要約統計量 (平均、標準偏差、分布)
- ▶ サンプルング
- ▶ グラフによる可視化
- ▶ 演習: gnuplot によるグラフ描画

データとばらつき

- ▶ データはばらつく
 - ▶ 真値に対して測定値がばらつく場合
 - ▶ 平均値を求めれば必要な値は得られる
 - ▶ (だが、値の信頼性を議論するにはばらつきの把握が必要)
 - ▶ 測定対象自体がばらついている場合
 - ▶ ばらつきを把握する必要

- ▶ ばらつきを把握する方法
 - ▶ 要約統計量
 - ▶ グラフによる可視化

要約統計量 (summary statistics)

標本の分布の特徴を要約して表す数値

- ▶ 位置を表す数値:
 - ▶ 平均 (mean)、中央値 (median)、最頻値 (mode)
- ▶ ばらつきを表す数値:
 - ▶ 範囲 (range)、分散 (variance)、標準偏差 (standard deviation)

位置を表す数値

- ▶ 平均 (mean):

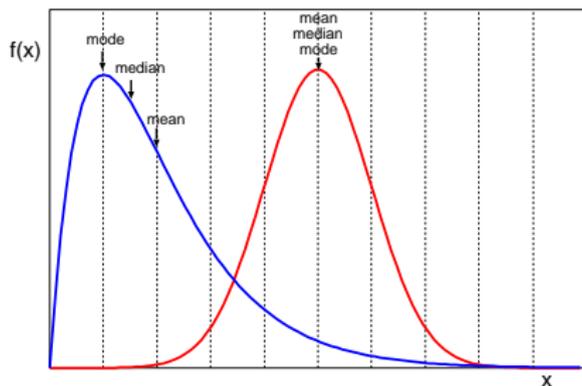
$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$$

- ▶ 中央値 (median): データの値をソートして中央にくる値

$$x_{median} = \begin{cases} x_{r+1} & m \text{ が奇数の場合, } m = 2r + 1 \\ (x_r + x_{r+1})/2 & m \text{ が偶数の場合, } m = 2r \end{cases}$$

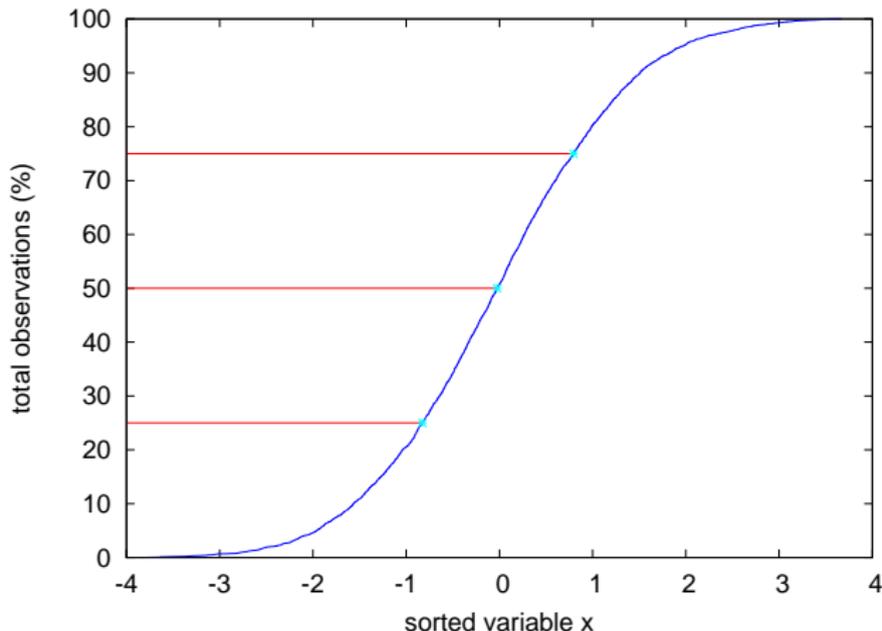
- ▶ 最頻値 (mode): 出現頻度が最も高い値

対称な分布であれば、これらは同一



パーセンタイル (percentiles)

- ▶ p th-percentile: 小さい方から数えて $p\%$ 目の値
 - ▶ median = 50th-percentile

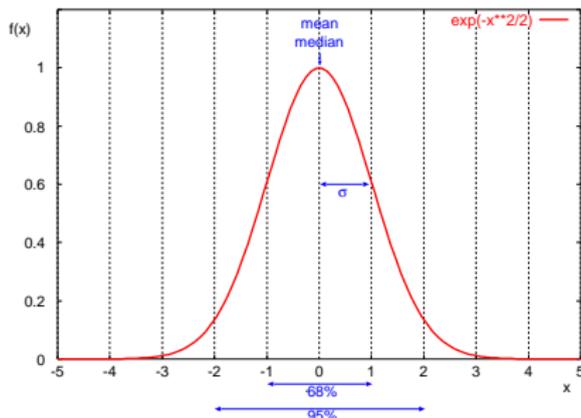


ばらつきを表す数値

- ▶ 範囲 (range): 最大値と最小値の差
- ▶ 分散 (variance):

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

- ▶ 標準偏差 (standard deviation): σ
 - ▶ 平均と同じ次元なので直接比較可能
 - ▶ 統計的なばらつきを示すのに最も良く使われる値
- ▶ 正規分布ではデータの 68%は ($mean \pm stdddev$)、95%は ($mean \pm 2stdddev$) の範囲に入る



分散の計算

分散 (variance):

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

この式のままで、一度平均を計算してから分散を計算する必要。
プログラミングでは、以下の式を使う方が簡単

$$\begin{aligned}\sigma^2 &= \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2 \\ &= \frac{1}{n} \sum_{i=1}^n (x_i^2 - 2x_i\bar{x} + \bar{x}^2) \\ &= \frac{1}{n} \left(\sum_{i=1}^n x_i^2 - 2\bar{x} \sum_{i=1}^n x_i + n\bar{x}^2 \right) \\ &= \frac{1}{n} \sum_{i=1}^n x_i^2 - 2\bar{x}^2 + \bar{x}^2 \\ &= \frac{1}{n} \sum_{i=1}^n x_i^2 - \bar{x}^2\end{aligned}$$

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

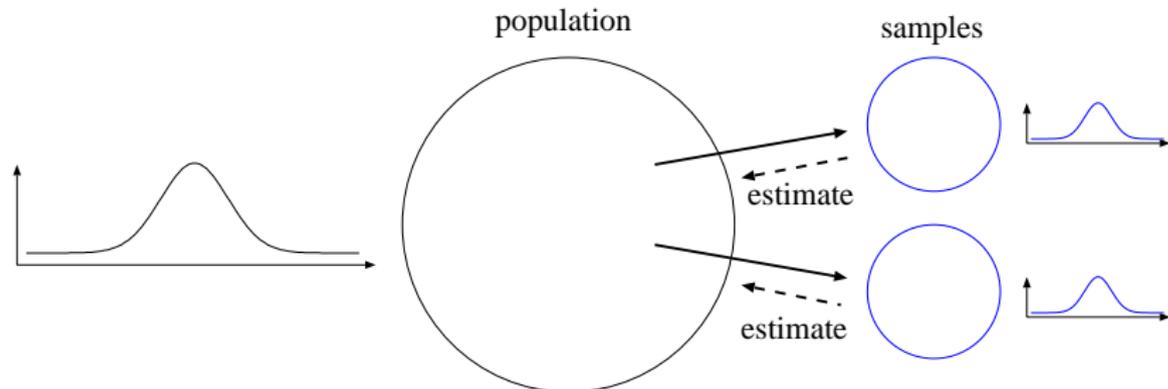
サンプリング: 標本と母集団

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づく

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 $N(\mu, \sigma/\sqrt{n})$
- ▶ 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)

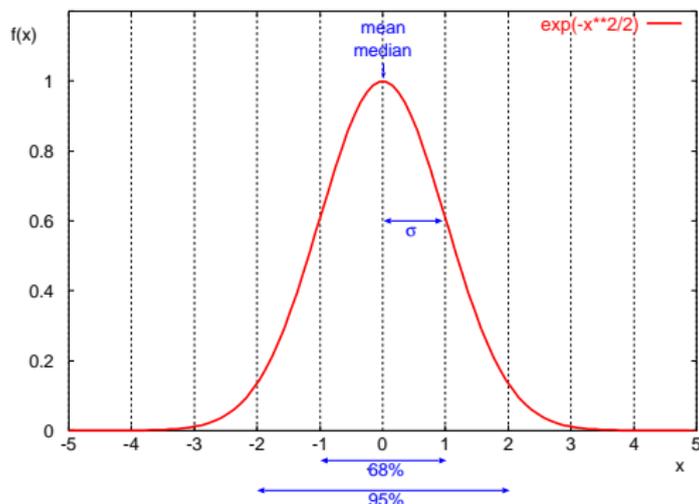
標準誤差: 標本平均の標準偏差 (SE)

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

- ▶ 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 $SE = \sigma / \sqrt{n}$ の正規分布となる

正規分布 (normal distribution)

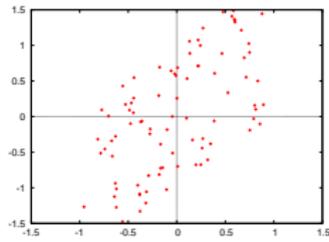
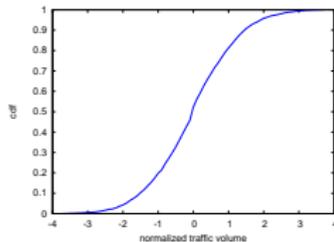
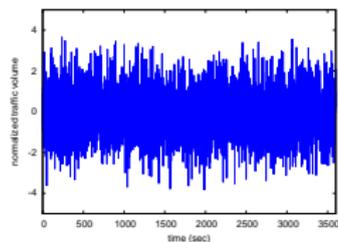
- ▶ つりがね型の分布、ガウス分布とも呼ばれる
- ▶ 2つの変数で定義: 平均 μ 、標準偏差 σ
- ▶ 乱数の和は正規分布に従う
- ▶ 標準正規分布: $\mu = 0, \sigma = 1$
- ▶ 正規分布ではデータの
 - ▶ 68%は ($mean \pm stddev$)
 - ▶ 95%は ($mean \pm 2stddev$) の範囲に入る



グラフ描画

データのばらつきを要約統計量だけから把握するのは難しい

直観的にデータの性質を把握するには、いくつかの統計的手法を用いてグラフを描画してみる



例: ある市民マラソンの完走時間分布

データ

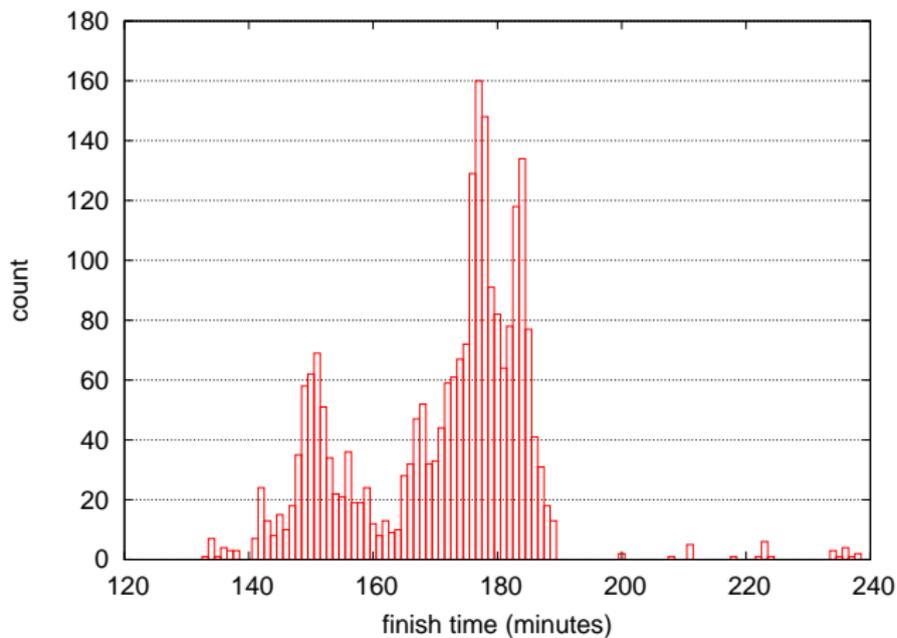
- ▶ sample data from a book: P. K. Janert “Gnuplot in Action”

```
# Minutes Count
133 1
134 7
135 1
136 4
137 3
138 3
141 7
142 24
...
```

完走者数:2,355 平均:171.3 分 標準偏差:14.1 中間値:176 分

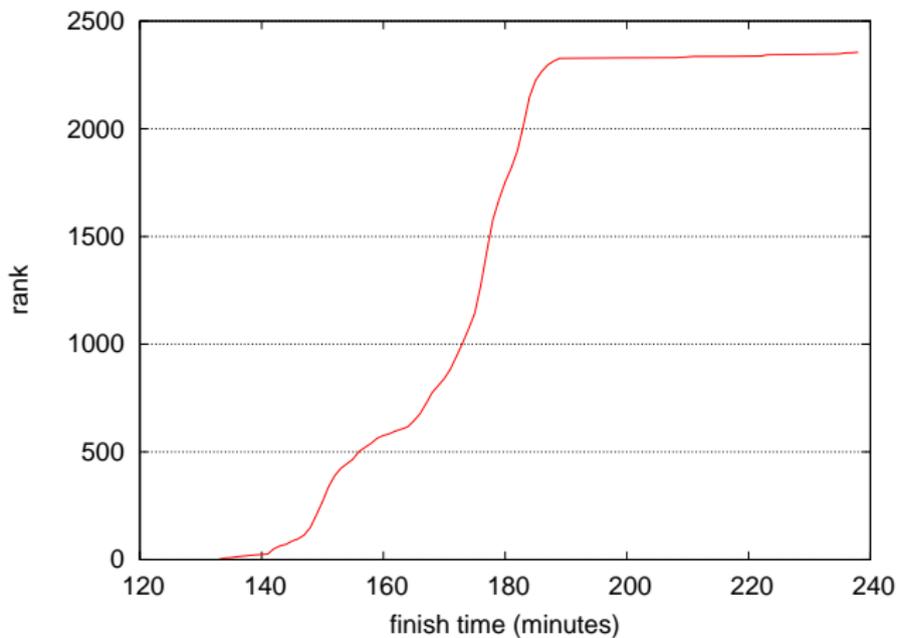
例: ある市民マラソンの完走時間分布 (2)

ヒストグラム



例: ある市民マラソンの完走時間分布 (3)

完走時間と順位分布



グラフ描画のガイドライン

読み手の立場にたって、分かり易いグラフを描画する

- ▶ XY 軸のラベルを明確に
- ▶ XY 軸の目盛りと単位を明確に
- ▶ 個々の直線曲線にもラベルを付ける
- ▶ 適切なフォントとサイズを使う
- ▶ 慣習に従う: 0 を起点にする、数学シンボルや略称の使用など
- ▶ ばらつきを示す (平均値だけでは不十分)
- ▶ グラフの範囲を適切か
- ▶ ひとつのグラフで多くを示さない
- ▶ 異なるデータを比較する場合は、適切な正規化を行う
- ▶ グラフ同士を比較する場合は、XY 軸のスケールを合わせる
- ▶ 技術系は円グラフや 3D 効果グラフは使わない
- ▶ 色を使う場合
 - ▶ 白黒印刷しても読めるように配慮
 - ▶ プロジェクタ投影も配慮 (例:黄色は避ける)

生データのグラフ化

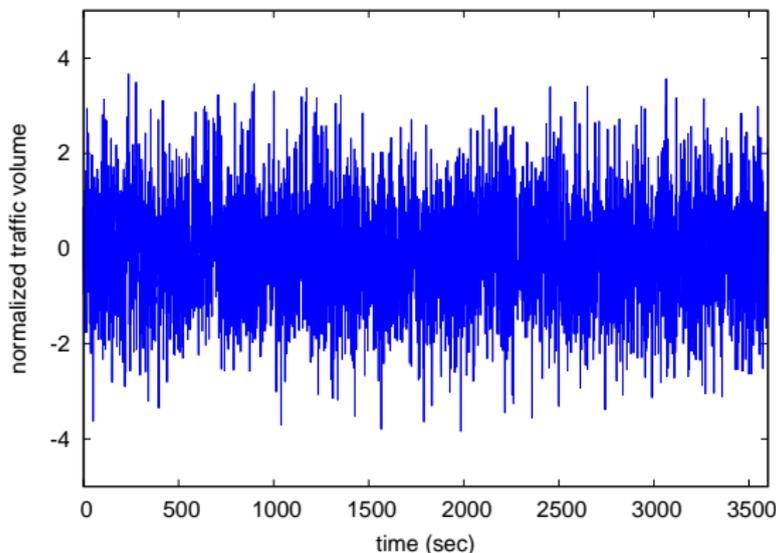
代表的なグラフ

- ▶ 時系列グラフ
- ▶ ヒストグラム
- ▶ 確率グラフ
- ▶ 散布図

時系列グラフ

変数の時間変化を見る

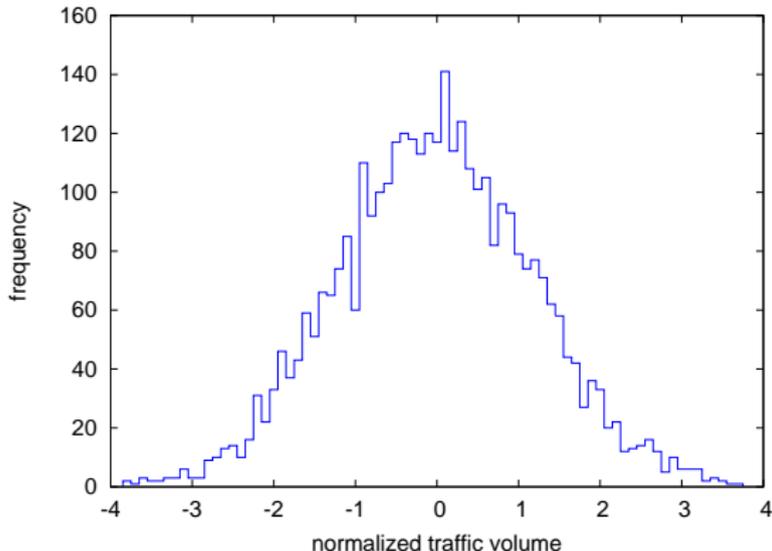
- ▶ X 軸に時間、Y 軸に変数値
- ▶ 時系列グラフから分かること
 - ▶ 位置の変化
 - ▶ ばらつきの変化
 - ▶ 外れ値の存在



ヒストグラム (1/2)

変数の分布の仕方を見る

- ▶ データを同じ幅のビンに分ける
- ▶ 各ビンのデータ数を数える
- ▶ X 軸:ビンの値 Y 軸:データ数



ヒストグラム (2/2)

ヒストグラムから分かる事

- ▶ 分布の中心 (位置)
- ▶ 分布の広がり
- ▶ 分布の偏り
- ▶ 外れ値の存在
- ▶ 複数のモードの存在 (山が複数あるか)

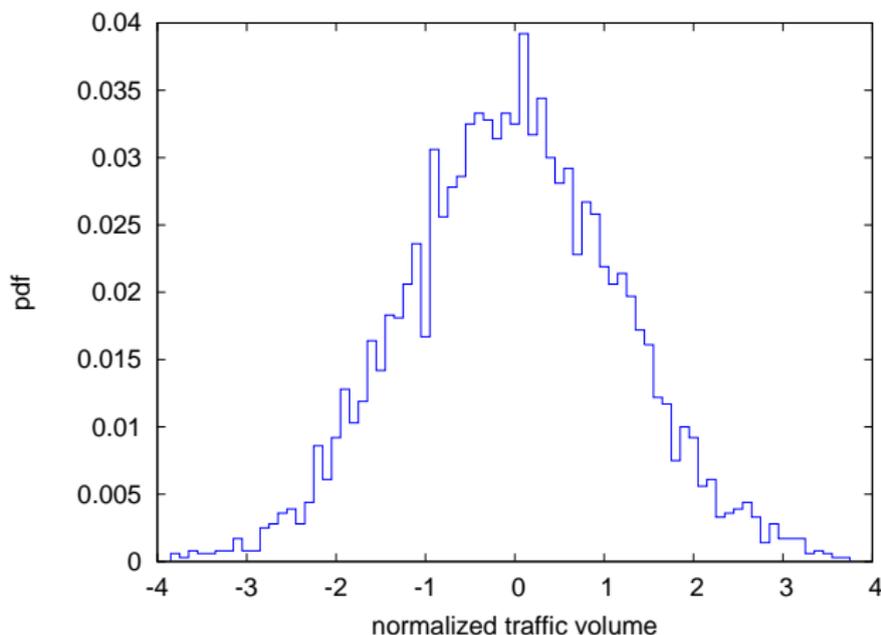
ヒストグラムの制約

- ▶ 適切なビン幅を選ぶ必要
 - ▶ 小さすぎると各ビンのサンプル数が足りなくなる
 - ▶ 大きすぎると分布の詳細が分からない
 - ▶ 偏りの大きい分布では適切なビン幅の選択は難しい
- ▶ 十分なサンプル数が必要

確率密度関数 (probability density function; pdf)

- ▶ 合計面積が 1 となるように出現数を正規化
 - ▶ 出現数を総データ数で割って相対度数にする
- ▶ 確率密度関数: 確率変数 X が x という値をとる確率

$$f(x) = P[X = x]$$



累積分布関数 (cumulative distribution function; cdf)

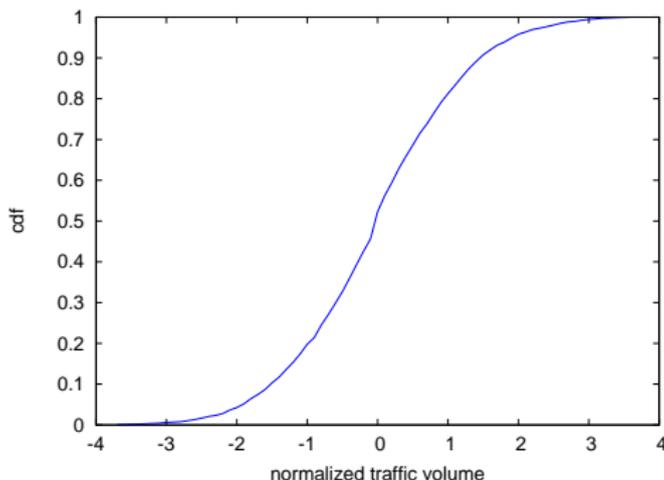
- ▶ 密度関数: x をいう値を観測する確率

$$f(x) = P[X = x]$$

- ▶ 累積分布関数: x 以下の値を観測する確率

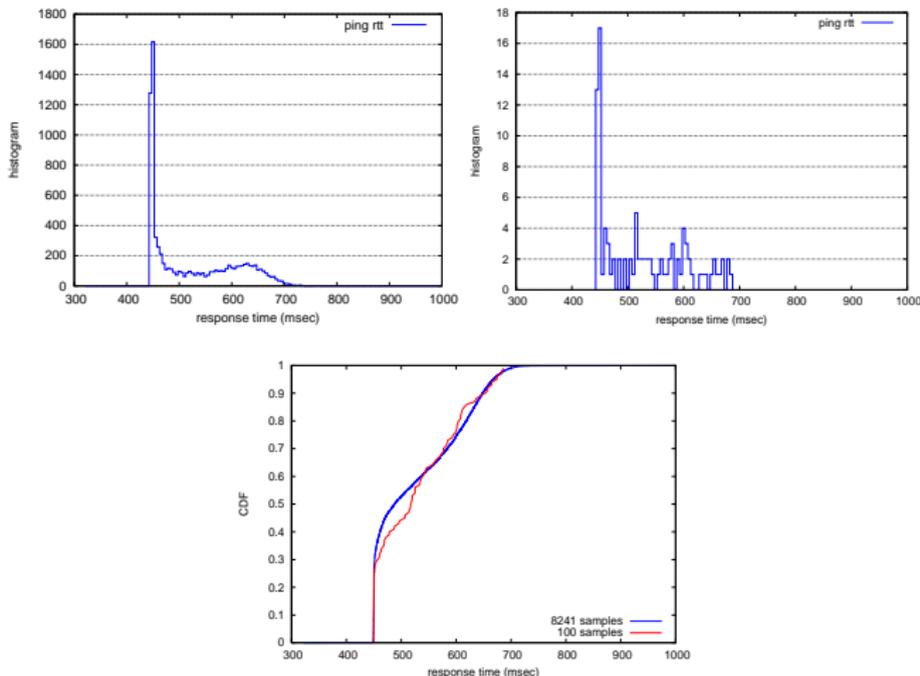
$$F(x) = P[X \leq x]$$

- ▶ 分布の偏りが大きい、サンプル数が少ない、外れ値が無視できない場合などは、ヒストグラムより有効



ヒストグラムと CDF の比較

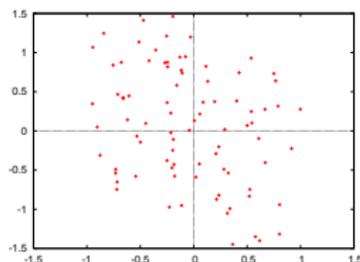
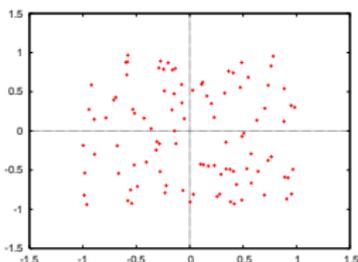
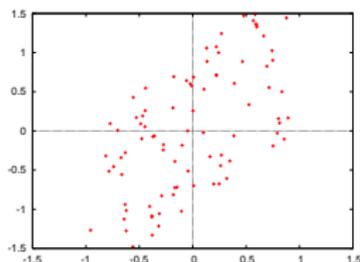
- ▶ CDF の場合、ビン幅やサンプル数不足を考慮しなくていい
- ▶ CDF の方が複数の分布を重ねて比較しやすい



(左) 元データ (右)100 サンプル (下)CDF

散布図 (scatter plots)

- ▶ 2 つの変数の関係を見るのに有効
 - ▶ X 軸: 変数 X
 - ▶ Y 軸: それに対応する変数 Y の値
- ▶ 散布図で分かる事
 - ▶ X と Y に関連があるか
 - ▶ 無相関、正の相関、負の相関
 - ▶ 外れ値の存在があるか



例: (左) 正の相関 0.7 (中) 無相関 0.0 (右) 負の相関 -0.5

グラフ描画ツール

- ▶ gnuplot
 - ▶ コマンドラインツール、スクリプトで自動化し易い
 - ▶ <http://gnuplot.info/>
- ▶ grace
 - ▶ 使い易い GUI
 - ▶ 細かい仕上げ調整が可能
 - ▶ <http://plasma-gate.weizmann.ac.il/Grace/>

前回の演習: ライン数をカウントするプログラム

引数ファイルのライン数をカウントする

```
filename = ARGV[0]
count = 0
file = open(filename)
while text = file.gets
  count += 1
end
file.close
puts count
```

count.rb というファイルにプログラムを書いて実行

```
$ ruby count.rb foo.txt
```

もう少し Ruby らしく書くと

```
#!/usr/bin/env ruby
count = 0
ARGV.each_line do |line|
  count += 1
end
puts count
```

演習: 要約統計量の計算

- ▶ 平均
- ▶ 標準偏差
- ▶ 中央値

- ▶ 市民マラソンのデータを使う: 出典 P. K. Janert “Gnuplot in Action”

<http://web.sfc.keio.ac.jp/~kjc/classes/sfc2013s-measurement/marathon.txt>

演習: 平均の計算

- ▶ 各行から、完走時間 (分) と人数を読み合計、最後に総数で割る

```
# regular expression to read minutes and count
re = /^(d+)\s+(d+)/
```

```
sum = 0 # sum of data
n = 0 # the number of data
ARGF.each_line do |line|
  if re.match(line)
    min = $1.to_i
    cnt = $2.to_i
    sum += min * cnt
    n += cnt
  end
end
```

```
mean = Float(sum) / n
```

```
printf "n:%d mean:%.1f\n", n, mean
```

```
% ruby mean.rb marathon.txt
n:2355 mean:171.3
```

演習: 標準偏差の計算

▶ アルゴリズム: $\sigma^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2$

```
# regular expression to read minutes and count
re = /^(d+)\s+(d+)/

data = Array.new
sum = 0 # sum of data
n = 0 # the number of data
ARGF.each_line do |line|
  if re.match(line)
    min = $1.to_i
    cnt = $2.to_i
    sum += min * cnt
    n += cnt
    for i in 1 .. cnt
      data.push min
    end
  end
end

mean = Float(sum) / n
sqsum = 0.0
data.each do |i|
  sqsum += (i - mean)**2
end

var = sqsum / n
stddev = Math.sqrt(var)
printf "n:%d mean:%.1f variance:%.1f stddev:%.1f\n", n, mean, var, stddev
```

```
% ruby stddev.rb marathon.txt
n:2355 mean:171.3 variance:199.9 stddev:14.1
```

演習: 標準偏差の計算の改良

▶ アルゴリズムを改良: $\sigma^2 = \frac{1}{n} \sum_{i=1}^n x_i^2 - \bar{x}^2$

```
# regular expression to read minutes and count
re = /^(d+)\s+(d+)/

sum = 0 # sum of data
n = 0 # the number of data
sqsum = 0 # su of squares
ARGV.each_line do |line|
  if re.match(line)
    min = $1.to_i
    cnt = $2.to_i
    sum += min * cnt
    n += cnt
    sqsum += min**2 * cnt
  end
end

mean = Float(sum) / n
var = Float(sqsum) / n - mean**2
stddev = Math.sqrt(var)

printf "n:%d mean:%.1f variance:%.1f stddev:%.1f\n", n, mean, var, stddev

% ruby stddev2.rb marathon.txt
n:2355 mean:171.3 variance:199.9 stddev:14.1
```

演習: 中央値の計算

- ▶ 各走者のタイムを配列に入れソート、中央値を取り出す

```
# regular expression to read minutes and count  
re = /^(d+)\s+(d+)/
```

```
data = Array.new
```

```
ARGF.each_line do |line|  
  if re.match(line)  
    min = $1.to_i  
    cnt = $2.to_i  
    for i in 1 .. cnt  
      data.push min  
    end  
  end  
end
```

```
data.sort! # just in case data is not sorted
```

```
n = data.length # number of array elements  
r = n / 2 # when n is odd, n/2 is rounded down  
if n % 2 != 0  
  median = data[r]  
else  
  median = (data[r - 1] + data[r])/2  
end
```

```
printf "r:%d median:%d\n", r, median
```

```
% ruby median.rb marathon.txt  
r:1177 median:176
```

演習: gnuplot

- ▶ gnuplot を使って簡単なグラフを書く

ヒストグラム

▶ 市民マラソンの完走タイムの分布

```
plot "marathon.txt" using 1:2 with boxes
```

グラフを見やすくする (右側)

```
set boxwidth 1
```

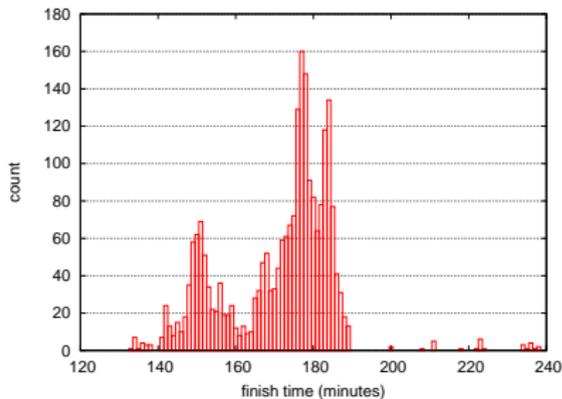
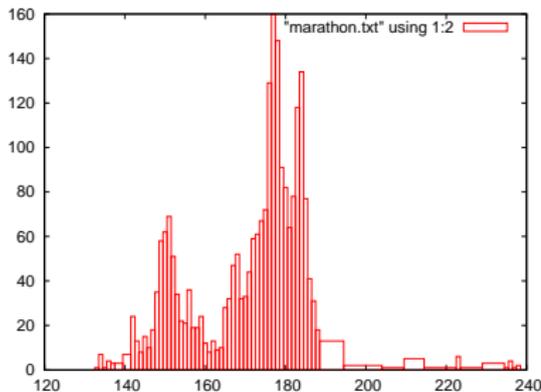
```
set xlabel "finish time (minutes)"
```

```
set ylabel "count"
```

```
set yrange [0:180]
```

```
set grid y
```

```
plot "marathon.txt" using 1:2 with boxes notitle
```



演習: 完走時間の CDF の作成

元データ:

```
# Minutes Count
133 1
134 7
135 1
136 4
137 3
138 3
141 7
142 24
...
```

累積度数を追加:

```
# Minutes Count CumulativeCount
133 1 1
134 7 8
135 1 9
136 4 13
137 3 16
138 3 19
141 7 26
142 24 50
...
```

演習: CDF (2)

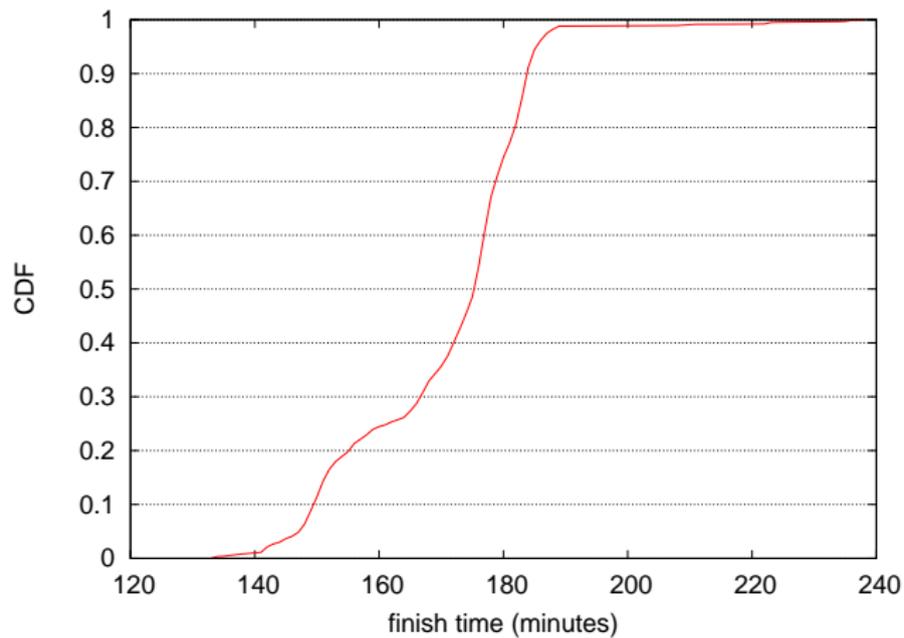
ruby code:

```
re = /^(\\d+)\\s+(\\d+)/
cum = 0
ARGF.each_line do |line|
  begin
    if re.match(line)
      # matched
      time, cnt = $~.captures
      cum += cnt.to_i
      puts "#{time}\\t#{cnt}\\t#{cum}"
    end
  end
end
```

gnuplot command:

```
set xlabel "finish time (minutes)"
set ylabel "CDF"
set grid y
plot "marathon-cdf.txt" using 1:($3 / 2355) with lines notitle
```

市民マラソンの完走時間 CDF



まとめ

第2回 データとばらつき

- ▶ 要約統計量 (平均、標準偏差、分布)
- ▶ サンプルング
- ▶ グラフによる可視化
- ▶ 演習: gnuplot によるグラフ描画

次回予定

第3回 データの収集と記録 (4/24)

- ▶ 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

参考文献

- [1] Ruby official site. <http://www.ruby-lang.org/>
- [2] gnuplot official site. <http://gnuplot.info/>
- [3] Mark Crovella and Balachander Krishnamurthy. *Internet measurement: infrastructure, traffic, and applications*. Wiley, 2006.
- [4] Pang-Ning Tan, Michael Steinbach and Vipin Kumar. *Introduction to Data Mining*. Addison Wesley, 2006.
- [5] Raj Jain. *The art of computer systems performance analysis*. Wiley, 1991.
- [6] Toby Segaran. (當山仁健 鴨澤眞夫 訳). 集合知プログラミング. オライリージャパン. 2008.
- [7] Chris Sanders. (高橋基信 宮本久仁男 監訳 岡真由美 訳). 実践パケット解析 第2版 — *Wireshark* を使ったトラブルシューティング. オライリージャパン. 2012.
- [8] あきみち、空閑洋平. インターネットのカタチ. オーム社. 2011.
- [9] 井上洋, 野澤昌弘. 例題で学ぶ統計的方法. 創成社, 2010.
- [10] 平岡和幸, 堀玄. プログラミングのための確率統計. オーム社, 2009.