Internet Measurement and Data Analysis (7)

Kenjiro Cho

2016-05-23

review of previous class

Class 6 Correlation (5/16)

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

today's topics

Class 7 Multivariate analysis

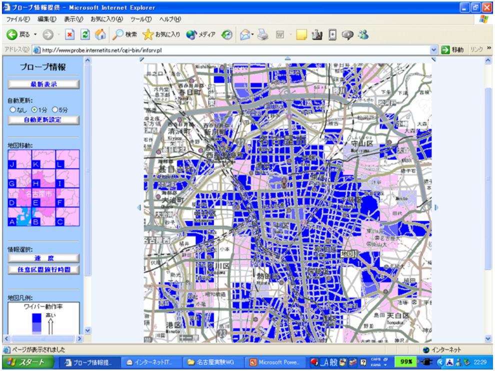
- Data sensing and GeoLocation
- Linear regression
- Principal Component Analysis
- exercise: linear regression and PCA

data sensing

- data sensing: collecting data from remote site
- it becomes possible to access various sensor information over the Internet
  - weather information, power consumption, etc.

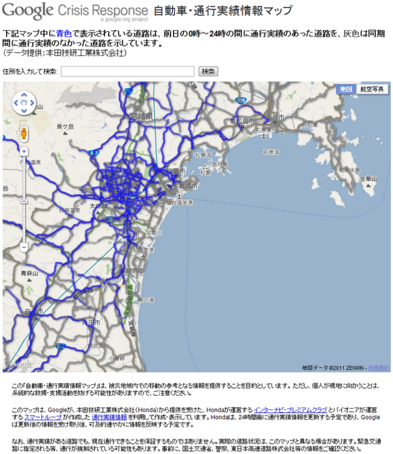
example: Internet vehicle experiment

- by WIDE Project in Nagoya in 2001
  - location, speed, and wiper usage data from 1,570 taxis
  - blue areas indicate high ratio of wiper usage, showing rainfall in detail



Japan Earthquake

- the system is now part of ITS
- usable roads info released 3 days after the quake
  - data provide by HONDA (TOYOTA, NISSAN)



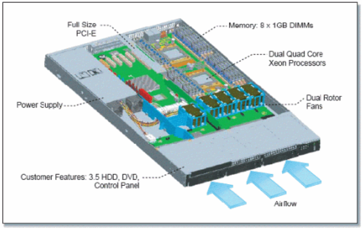
source: google crisis response

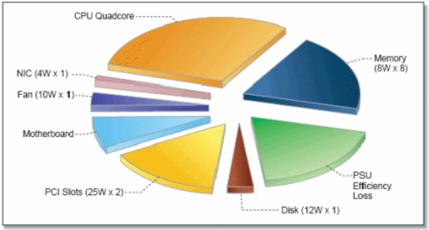
energy efficient technologies

- reduction in power consumption: issues in all technical fields
  - improving efficiency by intelligent control using sensor info
- from efficiency of individual equipment to efficiency of whole system
  - examples: PC servers and data centers

energy efficient PC servers

- intelligent control using sensor info within PC
  - temperature, voltage, power consumption, fan speed
- breakdown of PC server power consumption
  - CPU/memory: 50%
    - higher density, lower power, clock/voltage control
  - power supply: 20%
    - reduction in power loss (AC-DC, DC-DC)
  - IO: 20%
    - energy saving functions, energy efficient disks/SSD
  - cooling fans: 5%
    - better layout, air-flow design, optimized control

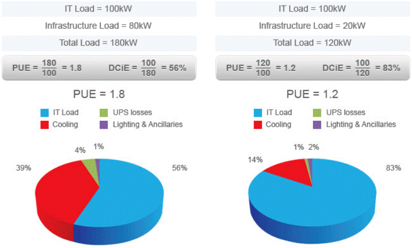




source: Intel Labs, 2006 and 2008

energy efficient data centers

- increasing power consumption by data centers with growing demands
  - contributed by cooling systems and power loss
- IT equipment: energy efficient equipment, use of servers with higher operating temperature
- cooling facility: spec reviews, air-flow/thermal-load design, energy efficient cooling equipment, free-air cooling
- power supply: loss reduction, high-voltage/DC power supply, energy efficient UPS, renewable energy
- total system design: adaptive control, human entry control, idle equipment shutdown



source: http://www.future-tech.co.uk/

GeoLocation Services

- to provide different services according to the user location
- map, navigation, timetable for public transportation
- search for nearby restaurants or other shops (for advertisement)
- possibilities for other services

example: 駅.Locky (Eki.Locky)

- train timetable service by Kawaguchi Lab, Nagoya University
  - popular app from a WiFi GeoLocation research project
- App for iPhone/Android
- automatically select the nearest station and show timetable
  - geo-location by GPS/WiFi
  - also collect WiFi access point info seen by the device
- countdown for the next train
  - can show timetalbe as well
- crowdsourcing: timetable database contributed by users

GPS (Global Positioning System)

- about 30 satellites for GPS
- originally developed for US military use
  - for civilian use, the accuracy was intentionally degraded to about 100m
  - in 2000, the accuracy was improved to about 10m by removing intentional noise
- wide variety of civilian usage
  - car navigation, mobile phones, digital cameras
- location measurement: locate the position by distances from 3 GPS satellites
  - GPS signal includes satellite position and time information
  - distance is calculated by the difference in the time in the signal
  - needs 4 satellites to calibrate the time of the receiver
  - the accuracy improves as more satellites are used
- limitations
  - needs to see satellites
  - initialization time to obtain initial positioning
- improvements: combine with accelerometers, gyro sensors, wifi geo-location

geo-location using access points

- a communication device knows its associated access point
  - an access point also knows associated devices
  - a device can receive signals from non-associated access points
- there exit services to get location information from access points
- can be used indoors
  - other approaches: sonic signals, visible lights
- can be combined with GPS to improve accuracy

measurement metrics of the Internet

measurement metrics

- link capacity, throughput
- delay
- jitter
- packet loss rate

methodologies

- active measurement: injects measurement packets (e.g., ping)
- passive measurement: monitors network without interfering in traffic
  - monitor at 2 locations and compare
  - infer from observations (e.g., behavior of TCP)
  - collect measurements inside a transport mechanism

delay measurement

- delay components
  - delay = propagation delay + queueing delay + other overhead
  - if not congested, delay is close to propagation deley
- methods
  - round-trip delay
  - one-way delay requires clock synchronization

  - average delay
  - max delay: e.g., voice communication requires $< 400ms$
  - jitter: variations in delay

some delay numbers

- packet transmission time (so called wire-speed)
  - 1500 bytes at 10Mbps: 1.2msec
  - 1500 bytes at 100Mbps: 120usec
  - 1500 bytes at 1Gbps: 12usec
  - 1500 bytes at 10Gbps: 1.2usec
- speed of light in fiber: about 200,000 km/s
  - 100km round-trip: 1 msec
  - 20,000km round-trip: 200msec
- satellite round-trip delay
  - LEO (Low-Earth Orbit): 200 msec
  - GEO (Geostationary Orbit): 600msec

packet loss measurement

packet loss rate

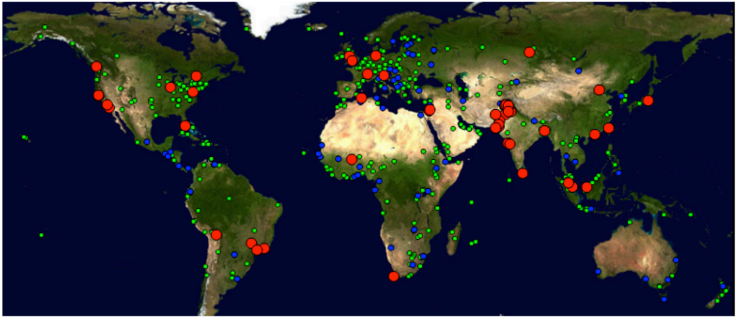
- loss rate is enough if packet loss is random...
- in reality,
  - bursty loss: e.g., buffer overflow
  - packet size dependency: e.g., bit error rate in wireless transmission

pingER project

- the Internet End-to-end Performance Measurement (IEPM) project by SLAC
- using ping to measure rtt and packet loss around the world
  - http://www-iepm.slac.stanford.edu/pinger/
  - started in 1995
  - over 600 sites in over 125 countries

pingER project monitoring sites

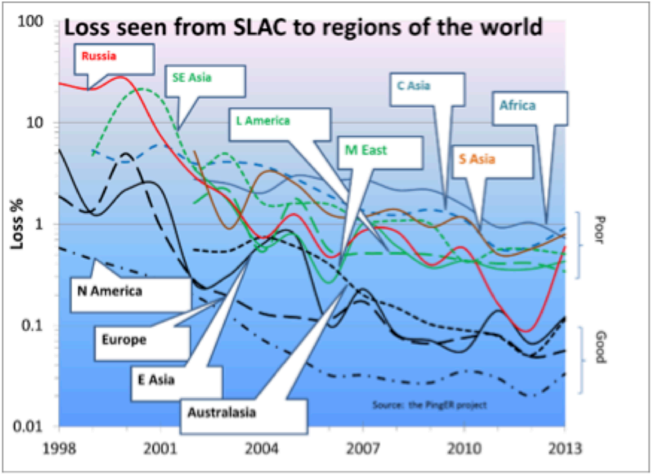
- monitoring (red), beacon (blue), remote (green) sites
  - beacon sites are monitored by all monitors



from pingER web site

pingER packet loss

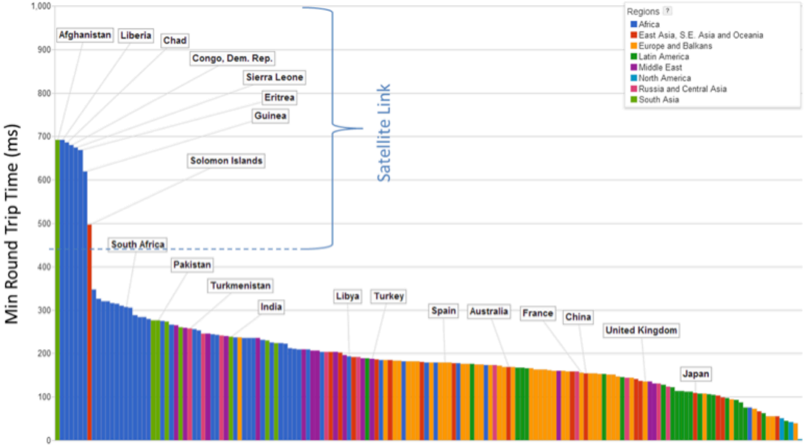
- packet loss observed from SLAC in the west coast
- exponential improvement in 15 years



from pingER web site

pinger minimum rtt

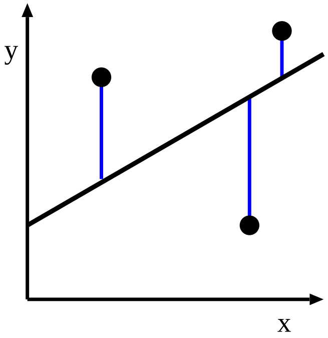
- minimum rtts observed from SLAC in the west coast

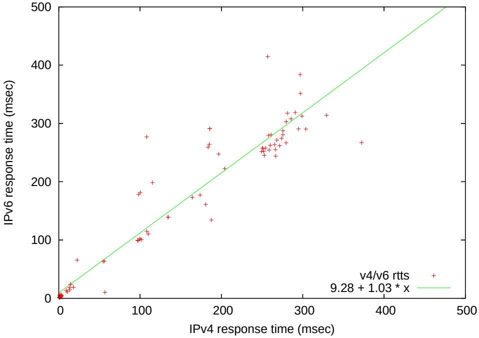


from pingER web site

linear regression

- fitting a straight line to data
  - least square method: minimize the sum of squared errors





least square method

a linear function minimizing squared errors

$$f(x) = b_0 + b_1 x$$

2 regression parameters can be computed by

$$b_1 = \frac{\sum xy - n\bar{x}\bar{y}}{\sum x^2 - n(\bar{x})^2}$$

$$b_0 = \bar{y} - b_1\bar{x}$$

where

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

$$\sum xy = \sum_{i=1}^{n} x_i y_i \qquad \sum x^2 = \sum_{i=1}^{n} (x_i)^2$$

a derivation of the expressions for regression parameters

The error in the ith observation: $e_i = y_i - (b_0 + b_1 x_i)$
For a sample of n observations, the mean error is

$$\bar{e} = \frac{1}{n}\sum_i e_i = \frac{1}{n}\sum_i (y_i - (b_0 + b_1 x_i)) = \bar{y} - b_0 - b_1\bar{x}$$

Setting the mean error to 0, we obtain: $b_0 = \bar{y} - b_1\bar{x}$
Substituting b_0 in the error expression:
$e_i = y_i - \bar{y} + b_1\bar{x} - b_1 x_i = (y_i - \bar{y}) - b_1(x_i - \bar{x})$
The sum of squared errors, SSE, is

$$SSE = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n}[(y_i - \bar{y})^2 - 2b_1(y_i - \bar{y})(x_i - \bar{x}) + b_1^2(x_i - \bar{x})^2]$$

$$\begin{aligned}\frac{SSE}{n} &= \frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})^2 - 2b_1\frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})(x_i - \bar{x}) + b_1^2\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2 \\ &= \sigma_y^2 - 2b_1\sigma_{xy}^2 + b_1^2\sigma_x^2\end{aligned}$$

The value of b_1, which gives the minimum SSE, can be obtained by differentiating this equation with respect to b_1 and equating the result to 0:

$$\frac{1}{n}\frac{d(SSE)}{db_1} = -2\sigma_{xy}^2 + 2b_1\sigma_x^2 = 0$$

That is: $b_1 = \frac{\sigma_{xy}^2}{\sigma_x^2} = \frac{\sum xy - n\bar{x}\bar{y}}{\sum x^2 - n(\bar{x})^2}$

principal component analysis; PCA

purpose of PCA

- convert a set of possibly correlated variables into a smaller set of uncorrelated variables

PCA can be solved by eigenvalue decomposition of a covariance matrix

applications of PCA

- demensionality reduction
  - sort principal components by contribution ratio, components with small contribution ratio can be ignored
- principal component labeling
  - find means of produced principal components

notes:

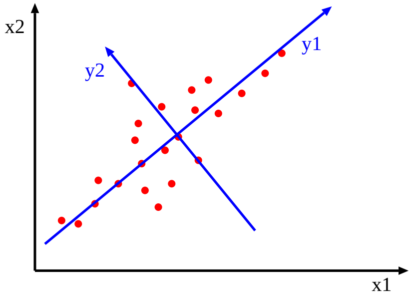
- PCA just extracts components with large variance
  - not simple if axes are not in the same unit
- a convenient method to automatically analyze complex relationship, but it does not explain the complex relationship

PCA: intuitive explanation

a view of cordinate transformation using a 2D graph

- draw the first axis (the 1st PCA axis) that goes through the centroid, along the direction of the maximal variability
- draw the 2nd axis that goes through the centroid, is orthogonal to the 1st axis, along the direction of the 2nd maximal variability
- draw the subsequent axes in the same manner

For example, “height” and “weight” can be mapped to “body size” and “slimness”. we can add “sitting height” and “chest measurement” in a similar manner



PCA (appendix)

principal components can be found as the eigenvectors of a covariance matrix.
let X be a d-demensional random variable. we want to find a $d \times d$ orthogonal transformation matrix P that converts X to its principal components Y.

$$\mathsf{Y} = \mathsf{P}^\top \mathsf{X}$$

solve this equation, assuming $cov(\mathsf{Y})$ being a diagonal matrix (components are independent), and P being an orthogonal matrix. $(\mathsf{P}^{-1} = \mathsf{P}^\top)$
the covariance matrix of Y is

$$\begin{aligned} cov(\mathsf{Y}) &= \mathsf{E}[\mathsf{Y}\mathsf{Y}^\top] = \mathsf{E}[(\mathsf{P}^\top\mathsf{X})(\mathsf{P}^\top\mathsf{X})^\top] = \mathsf{E}[(\mathsf{P}^\top\mathsf{X})(\mathsf{X}^\top\mathsf{P})] \\ &= \mathsf{P}^\top\mathsf{E}[\mathsf{X}\mathsf{X}^\top]\mathsf{P} = \mathsf{P}^\top cov(\mathsf{X})\mathsf{P} \end{aligned}$$

thus,

$$\mathsf{P}cov(\mathsf{Y}) = \mathsf{P}\mathsf{P}^\top cov(\mathsf{X})\mathsf{P} = cov(\mathsf{X})\mathsf{P}$$

rewrite P as a $d \times 1$ matrix:

$$\mathsf{P} = [\mathsf{P}_1, \mathsf{P}_2, \ldots, \mathsf{P}_d]$$

also, $cov(\mathsf{Y})$ is a diagonal matrix (components are independent)

$$cov(\mathsf{Y}) = \begin{bmatrix} \lambda_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_d \end{bmatrix}$$

this can be rewritten as

$$[\lambda_1\mathsf{P}_1, \lambda_2\mathsf{P}_2, \ldots, \lambda_d\mathsf{P}_d] = [cov(\mathsf{X})\mathsf{P}_1, cov(\mathsf{X})\mathsf{P}_2, \ldots, cov(\mathsf{X})\mathsf{P}_d]$$

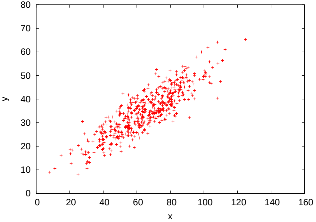
for $\lambda_i\mathsf{P}_i = cov(\mathsf{X})\mathsf{P}_i$, P_i is an eigenvector of the covariance matrix X
thus, we can find a transformation matrix P by finding the eigenvectors.

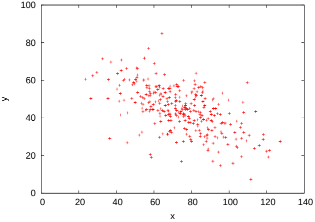
previous exercise: computing correlation coefficient

- compute correlation coefficient using the sample data sets
  - correlation-data-1.txt, correlation-data-2.txt

correlation coefficient

$$\rho_{xy} = \frac{\sigma_{xy}^2}{\sigma_x \sigma_y} = \frac{\sum_{i=1}^n x_i y_i - \frac{(\sum_{i=1}^n x_i)(\sum_{i=1}^n y_i)}{n}}{\sqrt{(\sum_{i=1}^n x_i^2 - \frac{(\sum_{i=1}^n x_i)^2}{n})(\sum_{i=1}^n y_i^2 - \frac{(\sum_{i=1}^n y_i)^2}{n})}}$$





data-1:r=0.87 (left), data-2:r=-0.60 (right)

script to compute correlation coefficient

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

# regular expression for matching 2 floating numbers
re = /([-+]?\d+(?:\.\d+)?)\s+([-+]?\d+(?:\.\d+)?)/

sum_x = 0.0     # sum of x
sum_y = 0.0     # sum of y
sum_xx = 0.0    # sum of x^2
sum_yy = 0.0    # sum of y^2
sum_xy = 0.0    # sum of xy
n = 0           # the number of data

ARGF.each_line do |line|
    if re.match(line)
      x = $1.to_f
      y = $2.to_f
      sum_x += x
      sum_y += y
      sum_xx += x**2
      sum_yy += y**2
      sum_xy += x * y
      n += 1
    end
end

r = (sum_xy - sum_x * sum_y / n) /
  Math.sqrt((sum_xx - sum_x**2 / n) * (sum_yy - sum_y**2 / n))

printf "n:%d r:%.3f\n", n, r
```

previous exercise 2: similarity

- compute similarity in data
  - data from "Programming Collective Intelligence" Section 2
  - movie rating scores of 7 people: scores.txt

```
% cat scores.txt
# A dictionary of movie critics and their ratings of a small set of movies
'Lisa Rose': 'Lady in the Water': 2.5, 'Snakes on a Plane': 3.5, 'Just My Luck': 3.0, 'Superman Returns':
'Gene Seymour': 'Lady in the Water': 3.0, 'Snakes on a Plane': 3.5, 'Just My Luck': 1.5, 'Superman Returns
'Michael Phillips': 'Lady in the Water': 2.5, 'Snakes on a Plane': 3.0, 'Superman Returns': 3.5, 'The Nigh
'Claudia Puig': 'Snakes on a Plane': 3.5, 'Just My Luck': 3.0, 'The Night Listener': 4.5, 'Superman Retur
'Mick LaSalle': 'Lady in the Water': 3.0, 'Snakes on a Plane': 4.0, 'Just My Luck': 2.0, 'Superman Returns
'Jack Matthews': 'Lady in the Water': 3.0, 'Snakes on a Plane': 4.0, 'The Night Listener': 3.0, 'Superman
'Toby': 'Snakes on a Plane':4.5,'You, Me and Dupree':1.0,'Superman Returns':4.0
```

score data

- simplistic example: data is too small
- summarized in the following table

```
#name: 'Lady in the Water' 'Snakes on a Plane' 'Just My Luck' 'Superman Returns
Lisa Rose:          2.5 3.5 3.0 3.5 3.0
Gene Seymour:       3.0 3.5 1.5 5.0 3.0
Michael Phillips:   2.5 3.0  -  3.5 4.0
Claudia Puig:        -  3.5 3.0 4.0 4.5
Mick LaSalle:       3.0 4.0 2.0 3.0 3.0
Jack Matthews:      3.0 4.0  -  5.0 3.0
Toby:                -  4.5  -  4.0  -
```

similarity computation

- create a similarity matrix using cosine similarity

```
% ruby similarity.rb scores.txt
Lisa Rose:        1.000 0.959 0.890 0.921 0.982 0.895 0.708
Gene Seymour:     0.959 1.000 0.950 0.874 0.962 0.979 0.783
Michael Phillips: 0.890 0.950 1.000 0.850 0.929 0.967 0.693
Claudia Puig:     0.921 0.874 0.850 1.000 0.875 0.816 0.695
Mick LaSalle:     0.982 0.962 0.929 0.875 1.000 0.931 0.727
Jack Matthews:    0.895 0.979 0.967 0.816 0.931 1.000 0.822
Toby:             0.708 0.783 0.693 0.695 0.727 0.822 1.000
```

similarity computation script (1/2)

```
# regular expression to read data
#       'name': 'title0': score0, 'title1': score1, ...
re = /'(.+?)':\s+(\S.*)/
name2uid = Hash.new     # keeps track of name to uid mapping
title2tid = Hash.new    # keeps track of title to tid mapping
scores = Hash.new       # scores[uid][tid]: score of title_id by user_id

# read data into scores[uid][tid]
ARGF.each_line do |line|
  if re.match(line)
    name = $1
    ratings = $2.split(",")

    if name2uid.has_key?(name)
      uid = name2uid[name]
    else
      uid = name2uid.length
      name2uid[name] = uid
      scores[uid] = {}  # create empty hash for title and score pairs
    end
    ratings.each do |rating|
      if rating.match(/'(.+?)':\s*(\d\.\d)/)
        title = $1
        score = $2.to_f
        if title2tid.has_key?(title)
          tid = title2tid[title]
        else
          tid = title2tid.length
          title2tid[title] = tid
        end
        scores[uid][tid] = score
      end
    end
  end
end
```

similarity computation script (2/2)

```
# compute cosine similarity between 2 users
def comp_similarity(h1, h2)
  sum_xx = 0.0  # sum of x^2
  sum_yy = 0.0  # sum of y^2
  sum_xy = 0.0  # sum of xy
  score = 0.0   # similarity score

  h1.each do |tid, score|
    sum_xx += score**2
    if h2.has_key?(tid)
      sum_xy += score * h2[tid]
    end
  end
  h2.each_value do |score|
    sum_yy += score**2
  end
  denom = Math.sqrt(sum_xx) * Math.sqrt(sum_yy)
  if denom != 0.0
    score = sum_xy / denom
  end
  return score
end

# create n x n matrix of similarities between users
n = name2uid.length
similarities = Array.new(n) { Array.new(n) }
for i in 0 .. n - 1
  printf "%-18s", name2uid.key(i) + ':'
  for j in 0 .. n - 1
    similarities[i][j] = comp_similarity(scores[i], scores[j])
    printf "%.3f ", similarities[i][j]
  end
  print "\n"
end
```

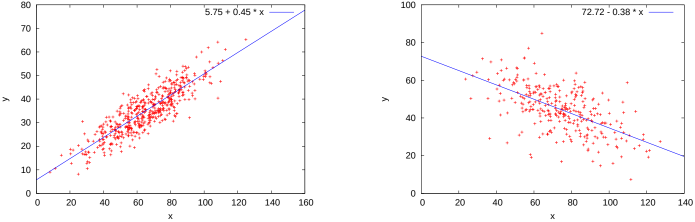
today's exercise: linear regression

- linear regression by the least square method
- use the data for the previous exercise
  - correlation-data-1.txt, correlation-data-2.txt

$$f(x) = b_0 + b_1 x$$

$$b_1 = \frac{\sum xy - n\bar{x}\bar{y}}{\sum x^2 - n(\bar{x})^2}$$

$$b_0 = \bar{y} - b_1 \bar{x}$$



data-1:r=0.87 (left), data-2:r=-0.60 (right)

script for linear regression

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

# regular expression for matching 2 floating numbers
re = /([-+]?\d+(?:\.\d+)?)\s+([-+]?\d+(?:\.\d+)?)/

sum_x = sum_y = sum_xx = sum_xy = 0.0
n = 0
ARGF.each_line do |line|
    if re.match(line)
      x = $1.to_f
      y = $2.to_f

      sum_x += x
      sum_y += y
      sum_xx += x**2
      sum_xy += x * y
      n += 1
    end
end

mean_x = Float(sum_x) / n
mean_y = Float(sum_y) / n
b1 = (sum_xy - n * mean_x * mean_y) / (sum_xx - n * mean_x**2)
b0 = mean_y - b1 * mean_x

printf "b0:%.3f b1:%.3f\n", b0, b1
```

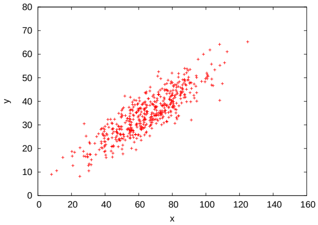
adding the least squares line to scatter plot

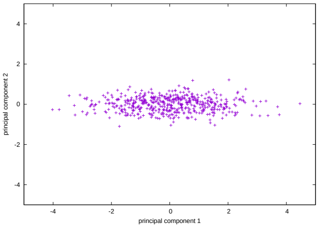
```
set xrange [0:160]
set yrange [0:80]

set xlabel "x"
set ylabel "y"

plot    "correlation-data-1.txt" notitle with points, \
        5.75 + 0.45 * x lt 3
```

today's exercise 2: PCA

- PCA: using the same datasets used for linear regression

```
$ ruby pca.rb correlation-data-1.txt
PC1:
eigenval: 1.86477
proportion: 0.93239
cumulative proportion: 0.93239
eigenvector: [0.7071067811865475, 0.7071067811865475]

PC2:
eigenval: 0.13523
proportion: 0.06761
cumulative proportion: 1.00000
eigenvector: [0.7071067811865475, -0.7071067811865475]
```





data-1:r=0.87 (left), pca plot (right)

PCA: with 3 variables

```
$ cat pca-data.txt
7 4 3
4 1 8
6 3 5
8 6 1
8 5 7
7 2 9
5 3 3
9 5 8
7 4 5
8 2 2
$ ruby pca.rb -p pca-data.txt
-0.542660  0.664959   0.035324
2.803897   -0.066207  0.348792
0.615631   0.306325   0.165059
-2.158526  0.958839   0.386086
-0.931052  -1.044819  0.360013
1.142388   -1.273946  0.471245
0.803082   1.261879   0.472342
-1.246820  -1.655638  -0.023007
-0.286027  -0.024512  0.186799
-0.199912  0.873118   -1.460164
```

```
$ ruby pca.rb pca-data.txt
PC1:
eigenval: 1.76877
proportion: 0.58959
cumulative proportion: 0.58959
eigenvector: [-0.642004576349, -0.686361641360, 0.341669169247]

PC2:
eigenval: 0.92708
proportion: 0.30903
cumulative proportion: 0.89862
eigenvector: [-0.384672291688, -0.0971303301343, -0.917928606687]

PC3:
eigenval: 0.30415
proportion: 0.10138
cumulative proportion: 1.00000
eigenvector: [-0.663217424343, 0.720745028589, 0.20166618906]
```

PCA script (1/4)

```
#!/usr/bin/env ruby
#
# usage: pca.rb [-p] datafile
#       input datafile: row: variables, column: observations
#       -p: convert input into principal components

# use matrix class for eigen vector computation
require 'matrix'
require 'optparse'

# nomarlize an array of array (m x n) into bb (m x n)
def normalize_matrix(aa)
  m = aa[0].length
  n = aa.length
  bb = Array.new(n) { Array.new(m) }    # normalized array of array

  for i in (0 .. m - 1)
    sum = 0.0           # sum of data
    sqsum = 0.0         # sum of squares
    for j in (0 .. n - 1)
      v = aa[j][i]
      sum += v
      sqsum += v**2
    end
    mean = sum / n
    stddev = Math.sqrt(sqsum / n - mean**2)
    for j in (0 .. n - 1)
      bb[j][i] = (aa[j][i] - mean) / stddev    # normalize
    end
  end
  bb    # return the new array of array
end
```

PCA script (2/4)

```
# make correlation matrix from data (array of array)
def make_corr_matrix(aa)
  m = aa[0].length
  n = aa.length
  corr_matrix = Array.new(m) { Array.new(m) }

  for i in (0 .. m - 1)
    for j in (0 .. m - 1)
      sum_x = 0.0
      sum_y = 0.0
      sum_xx = 0.0
      sum_yy = 0.0
      sum_xy = 0.0
      for k in (0 .. n - 1)
        x = aa[k][i]
        y = aa[k][j]
        sum_x += x
        sum_y += y
        sum_xx += x**2
        sum_yy += y**2
        sum_xy += x*y
      end
      cc = 0.0
      denom = (sum_xx - sum_x**2 / n) * (sum_yy - sum_y**2 / n)
      if denom != 0.0
        cc = (sum_xy - sum_x * sum_y / n) / Math.sqrt(denom)
      end
      corr_matrix[i][j] = cc
    end
  end
  corr_matrix
end
```

PCA script (3/4)

```
do_projection = false
OptionParser.new {|opt|
  opt.on('-p') {|v| do_projection = true}
  opt.parse!(ARGV)
}

# read data into input (array of array)
input = Array.new
ARGF.each_line do |line|
  values = line.split
  if values.length > 0
    row = Array.new
    values.each do |v|
      row.push v.to_f
    end
    input.push row
  end
end

corr_aa = make_corr_matrix(input) # create correlation matrix
corrmatrix = Matrix.rows(corr_aa) # convert array of array into matrix class

# compute the eigenvalues and eigenvectors
# eigensystem returns v: eigenvectors, d: diagonal matrix of eigenvalues,
#  v_inv: transposed matrix of v.  corrmatrix = v * d * v_inv
v, d, v_inv = corrmatrix.eigensystem

# returned vectors are sorted in increasing order of eigenvals.
# so, re-order eigenvalues and eigenvectors in decreasing order
eigenvals = Array.new^^I# array of eigenvalues
(d.column_size - 1).downto(0) do |i|
  eigenvals.push d[i,i]
end
eigenvectors = Matrix.columns(v.column_vectors.reverse)
```

PCA script (4/4)

```
if do_projection != true
  # show summaries
  eig_sum = 0.0
  eigenvals.each do |val|
    eig_sum += val
  end
  cum = 0.0  # cumulative of eigenvalues
  eigenvals.each_with_index do |val, i|
    printf "PC%d:\n", i + 1
    printf "eigenval: %.5f\n", val
    printf "proportion: %.5f\n", val / eig_sum
    cum += val
    printf "cumulative proportion: %.5f\n", cum / eig_sum
    print "eigenvector: "
    print eigenvectors.column(i).to_a
    print "\n\n"
  end
else
  # project the input into new coordinate
  # first, normalize the input and then convert it to matrix
  normalized = Matrix.rows(normalize_matrix(input))
  # projected data = eigenvec.T x normalized.T
  projected = eigenvectors.transpose * normalized.transpose

  projected.column_vectors.each do |vec|
    vec.each do |v|
      printf "%.6f\t", v
    end
    print "\n"
  end
end
```

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 2015
  - http://www.pseresults.com/events/741/results
  - the number of finishers: 21,554
- 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. discuss differences in finish time between male and female. what can you observe from the data?
  5. optional
     - other analysis of your choice (e.g., discussion on differences among age groups)
- submission format: a single PDF file including item 1-5
- submission method: upload the PDF file through SFC-SFS
- submission due: 2016-05-17

honolulu marathon data set

data format (compacted to fit in the slide)

Place	Chip Time	Number	Lname	Fname	Country	Category	Cat Place	Cat Total	5K	10K		40K	Gndr Place	Gndr Total	Pace
1	2:11:43	3	Kiprotich	Filex	KEN	MElite	1	5	16:07	31:40	...	2:04:48	1	11346	5:02
2	2:12:46	1	Chebet	Wilson	KEN	MElite	2	5	16:07	31:41	...	2:05:57	2	11346	5:04
3	2:13:24	8	Limo	Daniel	KEN	MElite	3	5	16:06	31:41	...	2:06:13	3	11346	5:06
4	2:15:27	6	Kwambai	Robert	KEN	MElite	4	5	16:08	31:41	...	2:07:29	4	11346	5:10
5	2:18:36	4	Mungara	Kenneth	KEN	MElite	5	5	16:07	31:40	...	2:09:42	5	11346	5:18
6	2:27:58	11	Neuschwander	Florian	DEU	M30-34	1	1241	17:46	34:50	...	2:20:31	6	11346	5:39
7	2:28:34	F1	Chepkirui	Joyce	KEN	WElite	1	7	16:53	33:21	...	2:20:56	1	10207	5:40
8	2:28:42	28803	Takahashi	Koji	JPN	M25-29	1	974	16:54	33:22	...	2:20:52	7	11346	5:41
9	2:28:55	F5	Karimi	Lucy	KEN	WElite	2	7	16:54	33:22	...	2:20:58	2	10207	5:41
10	2:29:44	F6	Ochichi	Isabella	KEN	WElite	3	7	16:53	33:22	...	2:21:46	3	10207	5:43

...

- Chip Time: finish time
- Number: bib number
- Category: MElite, WElite, M15-19, M20-24, ..., W15-29, W20-24, ...
  - note: 2 runners have "No Age" for Category, and num:18035 doesn't have cat/gender totals and its cat/gender placements are not reflected to the following entries
- Country: 3-letter country code: e.g., JPN, USA
- check the number of the total finishers when you extract the finishers

assignment 1: additional hints

- summary statistics: results can be in a table
- histograms:
  - X-axis: finish time (chip time) in 10min bin
  - Y-axis: the number of finishers for each bin
- CDF plot: (3 plots in a single figure)
  - X-axis: finish time
  - Y-axis: CDF [0:1]
- pages for the report: about 3-6 pages (source code not required)

sample code for extracting chip-time

```
# regular expression to read chiptime
re = /^\d+\s+(\d{1,2}:\d{2}:\d{2})\s+/

ARGF.each_line do |line|
  if re.match(line)
    puts "#{$1}"
  end
end
```

item 1: computing mean, standard deviation and median

- round off to minute (slightly different from using seconds)
- classify "No Age" using "Gender Total" (2 male finishers)

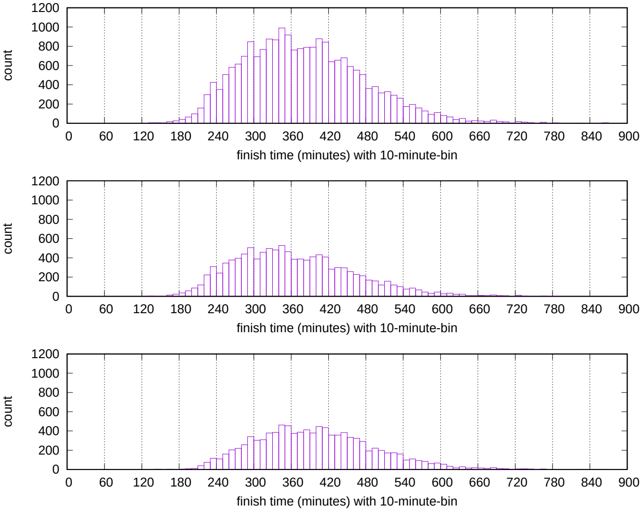
	n	mean	stddev	median
all	21,554	380.8	97.0	372
men	11,347	364.8	96.3	352
women	10,207	398.6	94.7	392

example script to extract data

```
# regular expression to read chiptime and category from honolulu.htm
re = /^\d+\s+(\d{1,2}:\d{2}:\d{2})\s+F?\d+\s+.*((?:[MW](?:Elite|\d{2}\-\d{2})|No Age))/
# alternative regular expression
#re = /^.{7} ?(\d{1,2}:\d{2}:\d{2}).{64}((?:[MW](?:Elite|\d{2}\-\d{2})|No Age))/

filename = ARGV[0]

open(filename, 'r') do |io|
  io.each_line do |line|
    if re.match(line)
      puts "#{$1}\t#{$2}"
    end
  end
end
```

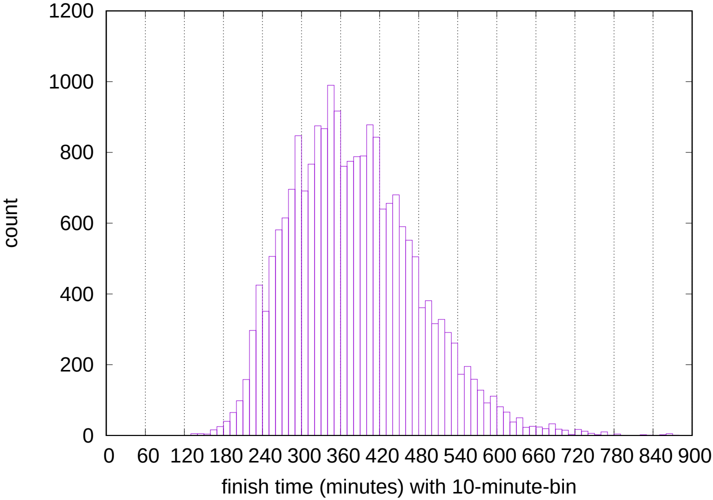
item 2: histograms for 3 groups

- 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

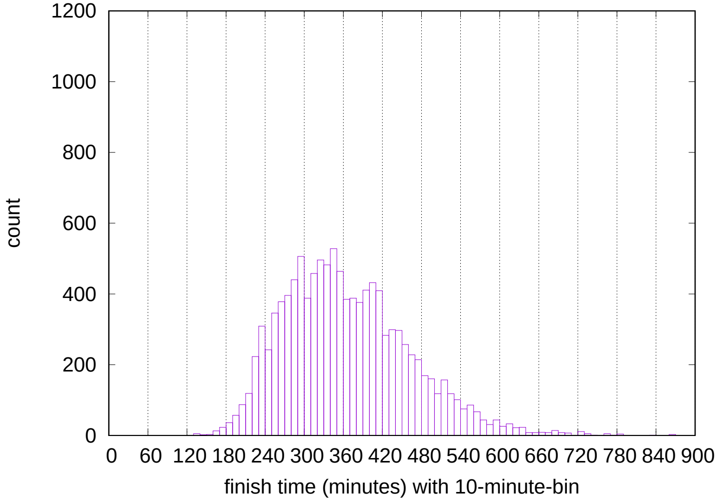


finish time histograms total(top) men(middle) women(bottom)

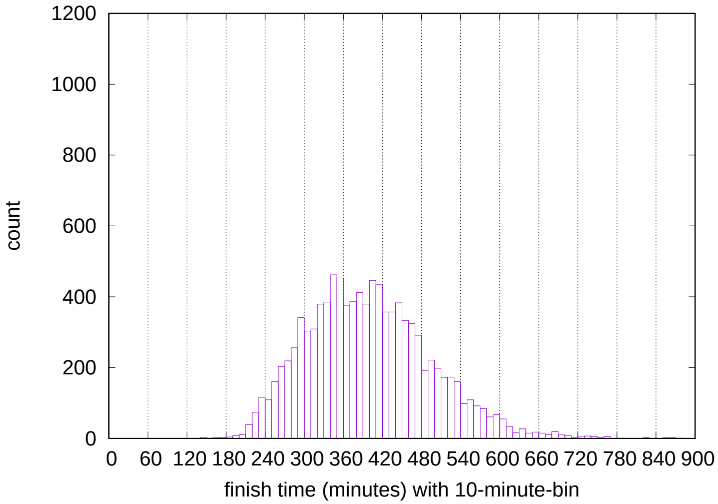
histograms for all



histograms for men

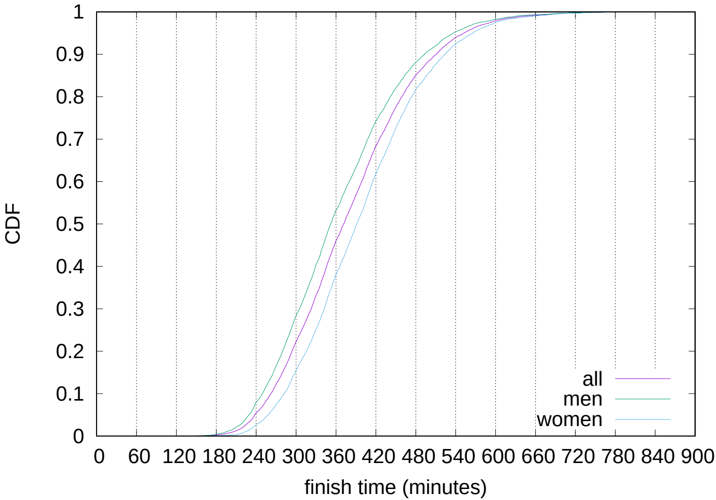


histograms for women



item 3: CDF of the finish time distributions of the 3 group

- plot 3 groups in a single graph



summary

Class 7 Multivariate analysis

- Data sensing and GeoLocation
- Linear regression
- Principal Component Analysis
- exercise: linear regression and PCA

next class

Class 8 Time-series analysis (5/30)

- Internet and time
- Network Time Protocol
- Time series analysis
- exercise: time-series analysis
- **assignment 2**