```lua
------------------------------------------------------------------------
---     __               __                   __                __
---  /\ \             /\ \                 /\ \              /\ \
---  \ \ \____    _ __\ \ \/'\      ___    \ \ \____    __    \_\ \
---   \ \ '__`\ /\`'__\ \ \ , <    /' _ `\   \ \ '__`\ /'__`\  /'_` \
---    \ \ \L\ \\ \ \/  \ \ \\`\  /\ \/\ \   \ \ \L\ \/\ \L\.\_/\ \L\ \
---     \ \_,__/ \ \_\   \ \_\ \_\\ \_\ \_\   \ \_,__/\ \__/.\_\ \___,_\
---      \/___/   \/_/    \/_/\/_/ \/_/\/_/    \/___/  \/__/\/_/\/__,_ /

---        .-------.
---        | Ba    | Bad <----.  planning= (better - bad)
---        |    56 |          |  monitor = (bad - better)
---        .-------.------.   |
---                | B    |   v
---                |    5 | Better
---                .------.
local b4={}; for k,_ in pairs(_ENV) do b4[k]=k end
local the, help = {}, [[

lua brknbad.lua [OPTIONS]
(c) 2022, Tim Menzies, BSD-2-Clause
Divide things. Show deltas between things.

OPTIONS:
  -cohen     -c cohen                   = .35
  -far       -F how far to seek poles = .9
  -keep      -k items to keep          = 256
  -minItems  -m min items in a rang e = .5
  -p         -p euclidean coefficient = 2
  -some      -S sample size for rows  = 512

OPTIONS, other:
  -dump      -d stackdump on error    = false
  -file      -f data file             = ../etc/data/auto93.csv
  -help      -h show help             = false
  -rnd       -r round numbers         = %5.2f
  -seed      -s random number seed    = 10019
  -todo      -t start-up action       = nothing
]]
local any, bestBin, bins, bins1, bootstrap, class, csv2egs, firsts, fmt, ish
local last, many, map, new, o, oo, per, push, quintiles, r, rnd, rnds, scottKnot
local selects, settings,slots, smallfx, sort, sum, thing, things, xplains
local Num, Sym, Egs, Bin, Cluster

-- ## Conventions:

-- ### Inference
-- - Recursive bi-clustering using random projections
-- - Inference via most distinguishing deltas between clusters
-- - Explanation = clustering + discretization

-- ### Data classes
-- - First row of data are names that describe each column.
-- - Names ending with `[+-]` are dependent goals to be minimized or maximized.
-- - Names ending with `!` are dependent classes.
-- - Dependent columns are `y` columns (the rest are independent `x` columns).
-- - Uppercase names are numeric (so the rest are symbolic).
-- - Names ending with `:` are columns to be skipped.
-- - Data is read as rows,  stored in a `Egs` instance.
-- - With a `Egs`, row columns are summarized into `Num` or `Sym` instances.

-- ### Code conventions
-- - No globals (so everything is `local`).
-- - Code 80 characters wide indent with two spaces.
-- - Format to be read a two-pages-per-page portrait pdf.
-- - Divide code into section and subsection headings (e.g using figlet)
-- - Sections are less than 120 lines long (one column in the pdf).
-- - No lines containing only the word `end` (unless marking the end of a
--   complex for loop or function).

-- ### Class conventions
-- - Spread class code across different sections (so don't overload reader
--   with all details, at one time).
-- - Show simpler stuff before complex stuff.
-- - Reserve `i` for `self` (to fit more code per line).
-- - Don't use inheritance (to simplify readability).
-- - Use polymorphism (using LUA's  delegation trick).
-- - Define an class of objects with `Thing=class"thing"` and
--   a `function:Thing(args)` creation method.
-- - Define instances with `new({slot1=value1,slot2=value2,...},Thing)`.
-- - Instance methods use `.`; e.g. `function Thing.show(i) ... end`.
-- - Class methods using `:`; e.g.  `Thing:new4strings`. Class methods
--   do things like instance creation or manage a set of instances.

-- ### Test suites (demos)
-- - Define start-up actions as `go` functions.
-- - In `go` functions, check for errors with `ok(test,mdf)`
--   (that updates an `fails` counter when not `ok`).

-- ### Top of file
-- - Trap known globals in `b4`.
-- - Define all locals at top-of-file (so everyone can access everything).
-- - Define options in a help string at top of file.
-- - Define command line options -h (for help); -s (for seeding random numbers)
--  `-t` (for startup actions, so `-t all` means "run everything").

-- ### End of file
-- - Using `settings`, parse help string to set options,
--   maybe updating from command-line.
-- - Using `go.main`, run the actions listed on command line.
-- - `go.main`  resets random number generator before running an action
-- - After everything else, look for `rogues` (any global not in `b4`)
-- - Finally, return the `fails` as the exit status of this code.

```

```lua
------------------------------------------------------------------------
---    _  _  _  ____  ____
---    |\/|  |  [__   |
---    |  |  |  ___]  |___

---     _ _   _. _|_|_  _
---    | | | (_|  |_| | _\

r=math.random
function ish(x,y,z) return math.abs(y -x ) < z end

---    |. __|_ _
---    ||_\ |  _\

function any(a)         return a[ math.random(#a) ] end
function firsts(a,b)    return a[1] < b[1] end
function last(a)        return a[ #a ] end
function many(a,n,  u)  u={}; for j=1,n do push(u,any(a)) end; return u end
function map(t,f, u)    u={};for _,v in pairs(t) do push(u,f(v)) end;return u end
function per(a,p)       return a[ (p*#a)//1 ] end
function push(t,x)      t[1 + #t] = x; return x end
function sort(t,f)      table.sort(t,f); return t end
function sum(t,f, n)
  f = f or function(x) return x end
  n=0; for _,v in pairs(t) do n = n + f(v) end; return n end

---     __|_ _.._  _    '~)  _|_|_ . _  _
---    _\ | | || |(_|   /_    | | |||| |(_|
---                  _|                  _|

function thing(x)
  x = x:match"^%s*(.-)%s*$"
  if x=="true" then return true elseif x=="false" then return false end
  return tonumber(x) or x end

function things(file,      x)
  local function cells(x,  t)
    t={}; for y in x:gmatch("([^,]+)") do push(t, thing(y)) end; return t end
  file = io.input(file)
  return function()
    x=io.read(); if x then return cells(x) else io.close(file) end end end

---    _|_|_ . _  _    '~)  __|_ _.._  _
---     | | |||| |(_|   /_  _\ | | || |(_|
---                  _|                  _|

fmt = string.format

function oo(t) print(o(t)) end

function o(t,  seen, u)
  if type(t)~="table" then return tostring(t) end
  seen = seen or {}
  if seen[t] then return "..." end
  seen[t] = t
  local function show1(x) return o(x, seen) end
  local function show2(k) return fmt(":%s %s",k,o(t[k],seen)) end
  u = #t>0 and map(t,show1) or map(slots(t),show2)
  return (t._is or "").."{"..table.concat(u," ").."}" end

function slots(t, u)
  u={};for k,v in pairs(t) do if tostring(k):sub(1,1)~="_" then push(u,k)end end
  return sort(u) end

function rnds(t,f) return map(t, function(x) return rnd(x,f) end) end
function rnd(x,f)
  return fmt(type(x)=="number" and (x~=x//1 and f or the.rnd) or "%s",x) end

---    |_  _ | _    _|_ _ _|_   '~)   _ _ _|__|_ . _  _  _
---    | |(/_||_)    | (/_><|    /_  _\(/_ |  | || |(_|_\
---           |                                     _|

function settings(help,    d)
  d={}
  help:gsub("\n ([-]([^%s]+))[%s]+(-[^%s]+)[^\n]*%s([^%s]+)",
    function(long,key,short,x)
      for n,flag in ipairs(arg) do
        if flag==short or flag==long then
          x = x=="false" and true or x=="true" and "false" or arg[n+1] end end
       d[key] = x==true and true or thing(x) end)
  if d.help then print(help) end
  return d end

---     _ _  _ _|_ _ _ |
---    (_(_)| | | | (_)|

local go, ok = {fails=0}
function ok(test,msg)
  print(test and "   PASS: "or "   FAIL: ",msg or "")
  if not test then
    go.fails = go.fails+1
    if the.dump then assert(test,msg) end end end

function go.main(todo,seed)
  for k,one in pairs(todo=="all" and slots(go) or {todo}) do
    if k ~= "main" and type(go[one]) == "function" then
      math.randomseed(seed)
      print(fmt(":%s",one))
      go[one]() end end
  for k,v in pairs(_ENV) do if not b4[k] then print("?",k,type(v)) end end  end

---     _ |_ . _  _ _|_ _
---    (_)|_)|(/_(_  | _\
---         L|

new = setmetatable
function class(s,   t)
  t={__tostring=o,_is=s or ""}; t.__index=t
  return new(t, {__call=function(_,...) return t.new(_,...) end}) end
```

```
------------------------------------------------------------------------
---     ___  ____ ___ ____    ____ _    ____ ____ ____ ____ ____
---     |  \ |__|  |  |__|    |    |    |__| [__  [__  |___ [__
---     |__/ |  |  |  |  |    |___ |___ |  | ___] ___] |___ ___]

Num, Sym, Egs = class"Num", class"Sym", class"Egs"

---      _ _  _  _ _|_ _
---     (_| (/_(_| |  (/_

function Sym:new(at,name)
    return new({at=at, name=name, most=0,n=0,all={}}, Sym) end

function Num:new(at,name)
    return new({at=at, name=name, _all={},
                w=(name or ""):find"-$" and -1 or 1,
                n=0, sd=0, mu=0, m2=0, lo=math.huge, hi=-math.huge}, Num) end

function Egs:new(names,  i,col)
  i = new({_all={}, cols={names=names, all={}, x={}, y={}}}, Egs)
  for at,name in pairs(names) do
    col = push(i.cols.all, (name:find"^[A-Z]" and Num or Sym)(at,name) )
    if not name:find":$" then
      if name:find"!$" then i.cols.class = col end
      push(name:find"[-+!]$" and i.cols.y or i.cols.x, col) end end
  return i end

function Egs:new4file(file,  i)
  for row in things(the.file) do
    if i then i:add(row) else i = Egs(row) end end
  return i end

---      _  _  _
---     (_(_)|_)\/
---          |  /

function Sym.copy(i) return Sym(i.at, i.name) end

function Num.copy(i) return Num(i.at, i.name) end

function Egs.copy(i,rows,    j)
  j = Egs(i.cols.names)
  for _,row in pairs(rows or {}) do j:add(row) end
  return j end

---          _  _|  _ _|_ _
---     |_||_)(_| (_|  |  (/_
---         |

function Egs.add(i,row)
  push(i._all,  row)
  for at,col in pairs(i.cols.all) do col:add(row[col.at]) end end

function Sym.add(i,x,inc)
  if x ~= "?" then
    inc = inc or 1
    i.n = i.n+inc
    i.all[x] = inc + (i.all[x] or 0)
    if i.all[x] > i.most then i.most, i.mode = i.all[x], x end end end

function Sym.sub(i,x,inc)
  if x ~= "?" then
    inc = inc or 1
    i.n = i.n - inc
    i.all[x] = i.all[x] - inc end end

function Num.add(i,x,_,    d,a)
  if x ~="?" then
    i.n   = i.n + 1
    d     = x - i.mu
    i.mu  = i.mu + d/i.n
    i.m2  = i.m2 + d*(x - i.mu)
    i.sd  = (i.m2<0 or i.n<2) and 0 or ((i.m2/(i.n - 1))^0.5)
    i.lo  = math.min(x, i.lo)
    i.hi  = math.max(x, i.hi)
    a     = i._all
    if     #a  < the.keep     then i.ok=false; push(a,x)
    elseif r() < the.keep/i.n then i.ok=false; a[r(#a)]=x end end end

function Num.sub(i,x,_,    d)
  if x ~="?" then
    i.n   = i.n - 1
    d     = x - i.mu
    i.mu  = i.mu - d/i.n
    i.m2  = i.m2 - d*(x - i.mu)
    i.sd  = (i.m2<0 or i.n<2) and 0 or ((i.m2/(i.n - 1))^0.5) end end

---      _        _  _
---     (_| |_|(/_|  \/
---       |/         /

function Egs.better(i,row1,row2)
  local s1, s2, n, a, b = 0, 0, #i.cols.y
  for _,col in pairs(i.cols.y) do
    a  = col:norm( row1[col.at] )
    b  = col:norm( row2[col.at] )
    s1 = s1 - 2.7183^(col.w * (a - b) / n)
    s2 = s2 - 2.7183^(col.w * (b - a) / n) end
  return s1 / n < s2 / n end

function Egs.betters(i,j,k)
  return i:better(j:mid(j.cols.all), k:mid(k.cols.all)) end

function Egs.mid(i,cols)
  return map(cols or i.cols.y, function(col) return col:mid() end) end

function Num.mid(i) return i.mu end
function Sym.mid(i) return i.mode end

function Num.div(i) return i.sd end
function Sym.div(i,  e)
  e=0; for _,n in pairs(i.all) do
         if n > 0 then e = e + n/i.n * math.log(n/i.n,2) end end
  return -e end

function Num.norm(i,x)
  return i.hi - i.lo < 1E-32 and 0 or (x - i.lo)/(i.hi - i.lo) end

function Num.all(i)
  if not i.ok then table.sort(i._all); i.ok=true end
  return i._all end

------------------------------------------------------------------------
---    ____ _    _  _ ____ ___ ____ ____
---    |    |    |  | [__   |  |___ |__/
---    |___ |___ |__| ___]  |  |___ |  \

Cluster=class"Cluster"
function Cluster:new(top,egs,      i,lefts,rights)
  egs = egs or top
  i   = new({egs=egs, top=top},Cluster)
  if #egs._all >= 2*(#top._all)^the.minItems then
    lefts, rights, i.left, i.right, i.mid, i.c = top:half(egs._all)
    if #lefts._all < #egs._all then
      i.lefts = Cluster(top, lefts)
      i.rights= Cluster(top, rights) end end
  return i end

function Cluster.leaf(i) return not (i.lefts or i.rights) end

function Cluster.show(i,   pre, front)
  pre = pre or ""
  local front = fmt("%s%s",pre,#i.egs._all)
  if   i:leaf()
  then print(fmt("%-20s%s",front, o(rnds(i.egs:mid(i.egs.cols.y)))))
  else print(front)
       if i.lefts  then i.lefts:show( "| "..pre)
       if i.rights then i.rights:show("| "..pre) end end end end

---     _ _  _  _| _  _ _    _  _ _ . _  __|_. _  _  _
---    | (_|| |(_|(_)| | |  |_)| (_)|(/_(_ |_|(_)| |_\
---                         |      L|

function Egs.half(i, rows)
  local project,far,some,left,right,c,lefts,rights
  far     = function(r,t)  return per(i:dists(r,t), the.far)[2] end
  project = function(r1,  a,b)
              a,b = i:dist(left,r1), i:dist(right,r1)
              return {(a^2 + c^2 - b^2)/(2*c), r1} end
  some    = many(rows,       the.some)
  left    = far(any(some), some)
  right   = far(left,      some)
  c       = i:dist(left,right)
  lefts,rights = i:copy(), i:copy()
  for n, projection in pairs(sort(map(rows,project),firsts)) do
    if n==#rows//2 then mid=row end
    (n <= #rows//2 and lefts or rights):add( projection[2] ) end
  return lefts, rights, left, right, mid, c  end

---     _|. __|_ _  _  _  _   . _    _| _ _|_ _
---    (_||_\ | (_|| |(_(/__\  || |  (_|(_| | (_|

function Egs.dists(i,r1,rows)
   return sort(map(rows,function(r2) return {i:dist(r1,r2),r2} end),firsts) end

function Egs.dist(i,row1,row2,    d)
  d = sum(i.cols.x, function(c) return c:dist(row1[c.at], row2[c.at])^the.p end)
  return (d/#i.cols.x)^(1/the.p) end

function Num.dist(i,a,b)
  if     a=="?" and b=="?" then return 1 end
  if     a=="?" then b=i:norm(b); a=b<.5 and 1 or 0
  elseif b=="?" then a=i:norm(a); b=a<.5 and 1 or 0
  else   a,b = i:norm(a), i:norm(b)  end
  return math.abs(a - b) end

function Sym.dist(i,a,b) return a=="?" and b=="?" and 1 or a==b and 0 or 1 end

--    $ lua brknbad.lua -t cluster
--
--    :cluster
--    398
--    | 199
--    | | 99
--    | | | 49
--    | | | | 24          {2542.50 15.68 26.25}
--    | | | | 25          {2408.48 17.72 35.20}
--    | | | 50
--    | | | | 25          {2432.12 16.04 28.80}
--    | | | | 25          {2504.20 16.52 30.80}
--    | | 100
--    | | | 50
--    | | | | 25          {2189.64 16.25 34.00} <== best
--    | | | | 25          {2261.56 16.24 28.80}
--    | | | 50
--    | | | | 25          {2309.24 16.74 26.00}
--    | | | | 25          {2194.60 16.10 26.00}
--    | 199
--    | | 99
--    | | | 49
--    | | | | 24          {3959.83 13.06 14.17}
--    | | | | 25          {4257.64 11.28 12.00} <== worst
--    | | | 50
--    | | | | 25          {3940.24 13.84 19.60}
--    | | | | 25          {4375.32 12.84 13.20}
--    | | 100
--    | | | 50
--    | | | | 25          {3220.32 17.40 21.20}
--    | | | | 25          {3259.04 16.39 22.00}
--    | | | 50
--    | | | | 25          {3189.96 16.32 20.00}
--    | | | | 25          {2504.56 16.56 23.20}
```

```
--------------------------------------------------------------------------
---     ___  _  ____  ____  ____  ____  ___  _  ___  ____
---    |   \ | [___  |     |___/ |___    |   |   /  |___
---    |___/ | ____] |____ |   \ |____   |   |  /__ |____

Bin=class"Bin"
function Bin:new(col,lo,hi,n,div)
  return new({col=col, lo=lo, hi=hi, n=n, div=div},Bin) end

function Bin.selects(i,row,  x)
  x = row[i.col.at]
  return x=="?" or i.lo==i.hi and x==i.lo or i.lo<=x and x<i.hi end

function Bin.show(i,negative)
  local x, big, s = i.col.name, math.huge
  if negative then
    if     lo==hi  then s=fmt("%s != %s",x,i.lo)
    elseif hi==big then s=fmt("%s < %s",x,i.lo)
    elseif lo==big then s=fmt("%s >= %s",x,i.hi)
    else                s=fmt("%s < %s and %s >= %s",x,i.lo,x,i.hi) end
  else
    if     lo==hi  then s=fmt("%s == %s",x,i.lo)
    elseif hi==big then s=fmt("%s >= %s",x,i.lo)
    elseif lo==big then s=fmt("%s < %s",x,i.hi)
    else                s=fmt("%s <= %s < %s",i.lo,x,i.hi) end end
  return s end

function Bin.distance2heaven(i, divs, ns)
  return ((1 - ns:norm(i.n))^2 + (0 - divs:norm(i.div))^2)^0.5 end

function Bin:best(bins)
  local divs,ns, distance2heaven = Num(), Num()
  function distance2heaven(bin) return {bin:distance2heaven(divs,ns),bin} end
  for _,bin in pairs(bins) do
    divs:add(bin.div)
    ns:add(  bin.ns) end
  return sort(map(bins, distance2heaven), firsts)[1][2]  end

---       _| .  _  _  _  _ _|_ . _  _      _      _  _   _
---     (_| | _\ (_ |  (/_  |  | /_ (/_   _\ \/ | | | _\
---                                           /

function Sym.bins(i,j)
  local xys= {}
  for x,n in pairs(i.all) do push(xys, {x=x,y="left", n=n}) end
  for x,n in pairs(j.all) do push(xys, {x=x,y="right",n=n}) end
  return Bin:new4Syms(i, Sym, xys) end

function Bin:new4Syms(col, yclass, xys)
  local out,all={}, {}
  for _,xy in pairs(xys) do
     all[xy.x] = all[xy.x] or yclass()
     all[xy.x]:add(xy.y, xy.n)  end
  for x,one in pairs(all) do push(out,Bin(col, x, x, one.n, one:div())) end
  return out end

---       _| .  _  _  _  _ _|_ . _  _     _         _  _   _
---     (_| | _\ (_ |  (/_  |  | /_ (/_   | | |_| | | | _\

function Num.bins(i,j)
  local xys, all = {}, Num()
  for _,n in pairs(i._all) do all:add(n); push(xys,{x=n,y="left"}) end
  for _,n in pairs(j._all) do all:add(n); push(xys,{x=n,y="right"}) end
  return Bin:new4Nums(i, Sym, sort(xys,function(a,b) return a.x < b.x end),
                      (#xys)^the.minItems, all.sd*the.cohen) end

function Bin:new4Nums(col, yclass, xys, minItems, cohen)
  local out,b4= {}, -math.huge
  local function bins1(lo,hi)
    local lhs, rhs, cut, div, xpect, xy = yclass(), yclass()
    for j=lo,hi do  rhs:add(xys[j].y) end
    div = rhs:div()
    for j=lo,hi do
      lhs:add(xys[j].y)
      rhs:sub(xys[j].y)
      if   lhs.n     > minItems and          -- enough items  (on left)
           rhs.n     > minItems and          -- enough items  (on right)
           xys[j].x ~= xys[j+1].x and        -- there is a break here
           xys[j].x  - xys[lo].x > cohen and -- not trivially small (on left)
           xys[hi].x - xys[j].x  > cohen     -- not trivially small (on right)
      then xpect = (lhs.n*lhs:div() + rhs.n*rhs:div()) / (lhs.n+rhs.n)
           if xpect < div then               -- cutting here simplifies things
             cut, div = j, xpect end end
    end
    if   cut
    then bins1(lo,    cut)
         bins1(cut+1, hi )
    else b4 = push(out, Bin(col, b4, xys[hi].x, hi-lo+1, div)).hi end
  end -----------------------------------------------
  bins1(1,#xys)
  out[#out].hi =  math.huge
  return out end
```

```
---          _ |  _  .  _
---     ><  |_)| (_| | | |
---         |

local xplain,xplains,selects,spanShow
function Egs.xplain(i,rows)
  local stop,here,left,right,lefts0,rights0,lefts1,rights1
  rows = rows or i._all
  here = {all=rows}
  stop = (#i._all)^the.minItems
  if #rows >= 2*stop then
    lefts0, rights0, here.left, here.right, here.mid, here.c  = half(i, rows)
    if #lefts0._all < #rows then
      cuts = {}
      for j,col in pairs(lefs0.col.x) do col:spans(rights0.col.x[j],cuts) end
      lefts1,rights1 = {},{}
      for _,row in pairs(rows) do
        push(selects(here.selector, row) and lefts1 or rights1, row) end
      if #lefts1  > stop then here.lefts  = xplain(i,lefts1) end
      if #rights1 > stop then here.rights = xplain(i,rights1) end end end
  return here end

function selects(span,row,    lo,hi,at,x)
  lo, hi, at = span.lo, span.hi, span.col.at
  x = row[at]
  if x=="?" then return true end
  if lo==hi then return x==lo else return lo <= x and x < hi end end

function xplains(i,format,t,pre,how,    sel,front)
  pre, how = pre or "", how or ""
  if t then
    pre=pre or ""
    front = fmt("%s%s%s %s",pre,how, #t.all, t.c and rnd(t.c) or "")
    if t.lefts and t.rights then print(fmt("%-35s",front)) else
      print(fmt("%-35s %s",front, o(rnds(mids(i,t.all),format))))
    end
    sel = t.selector
    xplains(i,format,t.lefts,  "|".. pre, spanShow(sel)..":")
    xplains(i,format,t.rights, "|".. pre, spanShow(sel,true) ..":") end end
```

```lua
---     __|_  _ _|_  _
---    _\ |  (_| |  _\

function quintiles(ts,width,  nums,out,all,n,m)
  width=width or 32
  nums=Num(); for _,t in pairs(ts) do
                for _,x in pairs(sort(t)) do add(nums,x) end end
  all,out = nums.all, {}
  for _,t in pairs(ts) do
     local s, where = {}
     where = function(n) return (width*nums:norm(n))//1 end
     for j = 1, width do s[j]=" " end
     for j = where(per(t,.1)), where(per(t,.3)) do s[j]="-" end
     for j = where(per(t,.7)), where(per(t,.9)) do s[j]="-" end
     s[where(per(t, .5))] = "|"
     push(out,{display=table.concat(s),
               data = t,
               pers = map({.1,.3,.5,.7,.9},
                          function(p) return rnd(per(t,p))end)}) end
  return out end

function smallfx(xs,ys,     x,y,lt,gt,n)
  lt,gt,n = 0,0,0
  if #ys > #xs then xs,ys=ys,xs end
  for _,x in pairs(xs) do
    for j=1, math.min(64,#ys) do
       y = any(ys)
       if y<x then lt=lt+1 end
       if y>x then gt=gt+1 end
       n = n+1 end end
  return math.abs(gt - lt) / n <= the.cliffs end

function bootstrap(y0,z0)
  local x, y, z, b4, yhat, zhat, bigger
  local function obs(a,b,    c)
    c = math.abs(a.mu - b.mu)
    return (a.sd + b.sd) == 0 and c or c/((x.sd^2/x.n + y.sd^2/y.n)^.5) end
  local function adds(t, num)
    num = num or Num(); map(t, function(x) add(num,x) end); return num end
  y,z    = adds(y0), adds(z0)
  x      = adds(y0, adds(z0))
  b4     = obs(y,z)
  yhat   = map(y._all, function(y1) return y1 - y.mu + x.mu end)
  zhat   = map(z._all, function(z1) return z1 - z.mu + x.mu end)
  bigger = 0
  for j=1,the.boot do
    if obs( adds(many(yhat,#yhat)),  adds(many(zhat,#zhat))) > b4
    then bigger = bigger + 1/the.boot end end
  return bigger >= the.conf end

--- xxx mid has to be per and
-- XXX implement same
-- XXX need tests for stats
function scottKnot(nums,      all,cohen)
  local mid = function (z) return z.some:mid()
  end -------------------------------
  local function summary(i,j,    out)
    out = copy( nums[i] )
    for k = i+1, j do out = out:merge(nums[k]) end
    return out
  end ---------------------------
  local function div(lo,hi,rank,b4,       cut,best,l,l1,r,r1,now)
    best = 0
    for j = lo,hi do
      if j < hi  then
        l   = summary(lo,  j)
        r   = summary(j+1, hi)
        now = (l.n*(mid(l) - mid(b4))^2 + r.n*(mid(r) - mid(b4))^2
              ) / (l.n + r.n)
        if now > best then
          if math.abs(mid(l) - mid(r)) >= cohen then
            cut, best, l1, r1 = j, now, copy(l), copy(r)
    end end end end
    if cut and not l1:same(r1,the) then
      rank = div(lo,    cut, rank, l1) + 1
      rank = div(cut+1, hi,  rank, r1)
    else
      for i = lo,hi do nums[i].rank = rank end end
    return rank
  end ------------------------------------------------------
  table.sort(nums, function(x,y) return mid(x) < mid(y) end)
  all   = summary(1,#nums)
  cohen = all.sd * the.cohen
  div(1, #nums, 1, all)
  return nums end
---------------------------------------------------------------------------
---     ____  ____
---    |  __ |    |
---    |___] |____|

function go.last()
  ok( 30 == last{10,20,30}, "lasts") end

function go.per(  t)
  t={};for i=1,100 do push(t,i*1000) end
  ok(70000 == per(t,.7), "per") end

function go.many(  t)
  t={};for i=1,100 do push(t,i) end; many(t,10) end

function go.sum(  t)
  t={};for i=1,100 do push(t,i) end; ok(5050==sum(t),"sum")end

function go.sample(   m,n)
  m,n = 10^5,Num(); for i=1,m do n:add(i) end
  for j=.1,.9,.1 do
    print(j,per(n:all(),j),ish(per(n:all(),j),m*j,m*0.05)) end end

function go.sym(  s)
  s=Sym(); map({1,1,1,1,2,2,3}, function(x) s:add(x) end)
  ok(ish(s:div(),1.378, 0.001), "ent") end

function go.num( n)
  n=Num(); map({10, 12, 23, 23, 16, 23, 21, 16}, function(x) n:add(x) end)
  print(n:div())
  ok(ish(n:div(),5.2373, .001), "div") end

function go.nums( num,t,b4)
  b4,t,num={},{},Num()
  for j=1,1000 do push(t,100*r()*j) end
  for j=1,#t  do
    num:add(t[j])
    if j%100==0 then    b4[j] =  fmt("%.5f",num:div()) end end
  for j=#t,1,-1 do
    if j%100==0 then ok(b4[j] == fmt("%.5f",num:div()),"div"..j) end
    num:sub(t[j]) end end

function go.syms( t,b4,s,sym)
  b4,t,sym, s={},{},Sym(), "I have gone to seek a great perhaps."
  t={}; for j=1,20 do s:gsub('.',function(x) t[#t+1]=x end) end
  for j=1,#t  do
    sym:add(t[j])
    if j%100==0 then    b4[j] =  fmt("%.5f",sym:div()) end end
  for j=#t,1,-1 do
    if j%100==0 then ok(b4[j] == fmt("%.5f",sym:div()),"div"..j) end
    sym:sub(t[j]) end
  end

function go.loader(  num)
  for row in things(the.file) do
    if num then num:add(row[1]) else num=Num() end end
  ok(ish(num.mu, 5.455,0.001),"loadmu")
  ok(ish(num.sd, 1.701,0.001),"loadsd") end

function go.egsShow(  e)
  ok(Egs{"name","Age","Weigh-"},"can make Egs?") end

function go.egsHead( )
  ok(Egs({"name","age","Weight!"}).cols.x,"Egs")  end

function go.egs(   egs)
  egs = Egs:new4file(the.file)
  ok(ish(egs.cols.x[1].mu, 5.455,0.001),"loadmu")
  ok(ish(egs.cols.x[1].sd, 1.701,0.001),"loadsd") end

function go.dist(  ds,egs,one,d1,d2,d3,r1,r2,r3)
  egs = Egs:new4file(the.file)
  one = egs._all[1]
  ds={};for j=1,20 do
        push(ds,egs:dist(any(egs._all), any(egs._all))) end
  oo(rnds(sort(ds),"%5.3f"))
  for j=1,10 do
    r1,r2,r3 = any(egs._all), any(egs._all), any(egs._all)
    d1=egs:dist(r1,r2)
    d2=egs:dist(r2,r3)
    d3=egs:dist(r1,r3)
    ok(d1<= 1 and d2 <= 1 and d3 <= 1 and d1>=0 and d2>=0 and d3>=0 and
       egs:dist(r1,r2) == egs:dist(r2,r1) and
       egs:dist(r1,r1) == 0               and
       d3 <= d1+d2,                       "dist"..j)  end end

function go.half(  egs,lefts,rights)
  egs = Egs:new4file(the.file)
  lefts, rights = egs:half(egs._all)
  oo(rnds(egs:mid()))
  print(egs:betters(lefts, rights))
  print(egs:betters(rights, lefts))
  oo(rnds(lefts:mid()))
  oo(rnds(rights:mid())) end

function go.cluster(   cl)
  Cluster(Egs:new4file(the.file)):show()  end

---------------------------------------------------------------------------
the = settings(help)
go.main(the.todo, the.seed)
os.exit(go.fails)


---               .---------.
---               |         |
---             -= _________ =-
---                ___   ___
---               |   )=(   |
---                ---   ---
---
---                  ###
---                #  =  #            "This ain't chemistry.
---                #######             This is art."
---                  ###

```