```lua
#!/usr/bin/env lua
local your, our={}, {go={}, no={}, b4={}, help=[[
duo.lua [OPTIONS]
(c)2022 Tim Menzies, MIT license (2 clause)
Data miners using/used by optimizers.
Understand N items after log(N) probes, or less.

  -file   ../../data/auto93.csv
  -ample  512
  -far    .9
  -best   .5
  -help   false
  -dull   .5
  -rest   3
  -seed   10019
  -Small .35
  -rnd    %.2f
  -task   -
  -p      2]]}

for k,_ in pairs(_ENV) do our.b4[k] = k end
local all,any,bsearch,firsts,fmt,new,many,map,o,push
local rows,seconds,slots,sort,thing,things
local EGS, NUM, RANGE, SYM = {},{},{},{}
-- ------------------------------------------------------------------
function RANGE.new(k,col,lo,hi,b,B,r,R)
  return new(k,{col=col,lo=lo,hi=hi or lo,b=b,B=B,r=r,R=R}) end

function RANGE.__lt(i,j) return i:val() < j:val() end

function RANGE.__tostring(i)
  if i.lo == i.hi        then return fmt("%s == %s", i.col.txt, i.lo) end
  if i.lo == -math.huge then return fmt("%s < %s",   i.col.txt, i.hi) end
  if i.hi ==  math.huge then return fmt("%s >= %s", i.col.txt, i.lo) end
  return fmt("%s <= %s < %s", i.lo, i.col.txt, i.hi) end

function RANGE.val(i,    z,B,R)
  z=1E-31; B,R = i.B+z, i.R+z; return (i.b/B)^2/( i.b/B + i.r/R) end

function RANGE.selects(i,row,     x)
  x=row.has[col.at]; return x=="?" or i.lo<=x and x<i.hi end
-- ------------------------------------------------------------------
function NUM.new(k,at,s)
  return new(k,{at=at,txt=s,w=s:find"-" and -1 or 1,_has={},
                ok=false, lo=math.huge, hi=-math.huge}) end

function NUM.add(i,x)
  if x ~= "?" then
    i.ok = false
    push(i._has, x)
    if x < i.lo then i.lo = x end
    if x > i.hi then i.hi = x end end
  return x end

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

function NUM.has(i)
  if not i.ok then sort(i._has); i.ok=true end; return i._has end

function NUM.norm(i,x)
  return i.hi - i.lo<1E-9 and 0 or (x - i.lo)/(i.hi - i.lo) end

-- compare to old above
function NUM.ranges(i,j,lo,hi)
  local z,is,js,lo,hi,m0,m1,m2,n0,n1,n2,step,most,best,r1,r2
  is,js   = i:has(), j:has()
  lo,hi   = lo or is[1], hi or is[#is]
  gap,max = (hi - lo)/16, -1
  if hi-lo < 2*gap then
    z      = 1E-32
    m0, m2 = bsearch(is, lo), bsearch(is, hi+z)
    n0, n2 = bsearch(js, lo), bsearch(js, hi+z)
    --                    col,lo hi,b     B   r     R
    best    = nil
    for mid in lo,hi,gap do
      if mid > lo and k < hi then
        m1 = bsearch(is, mid+z)
        n1 = bsearch(js, mid+z)
        --              col,  lo hi, b     B   r         R
        r1 = RANGE:new(i,     lo,mid,m1-m0,i.n,m2-(m1+1),j.n)
        r2 = RANGE:new(i, mid+z,hi, n1-n0,i.n,n2-(n1+1),j.n)
        if r1:val() > max then best, max = r1, r1:val() end
        if r2:val() > max then best, max = r2, r2:val() end end end end
  if   best
  then return i:ranges(j, best.lo, best.hi)
  else return RANGE:new(i,  lo,hi,m2-m0,i.n,n2-n0,j.n) end end

-- ------------------------------------------------------------------
function SYM.new(k,at,s)
  return new(k,{at=at,txt=s,_has={}}) end

function SYM.add(i,x)
  if x ~= "?" then i._has[x] = 1+(i.has[x] or 0) end
  return x end

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

function SYM.has(i)  return i.has end

function SYM.ranges(i,j)
  return map(i._has,
      function(x,n) return RANGE:new(i,x,x,n,i.n,(j._has[k] or 0),j.n) end) end
-- ------------------------------------------------------------------
function EGS.new(k)
  return new(k,{_rows={}, cols=nil, x={},  y={}}) end

function EGS.add(i,t)
  local add,now,where = function(col) return col:add(t[col.at]) end
  if   i.cols
  then push(i._rows, map(i.cols, add))
  else i.cols = {}
       for n,x in pairs(t) do
         now = (x:find"^[A-Z]" and NUM or SYM):new(n,x)
         push(i.cols, now)
         if not x:find":" then
           where = (x:find"+" or x:find"-") and i.y or i.x
           push(where, now) end end end end

function EGS.clone(i,inits,    j)
  j = EGS:new()
  j:add(map(i.cols, function(col) return col.txt end))
  for _,row in pairs(inits or {}) do j = j:add(row) end
  return j end

function EGS.cluster(i,top,lvl,         tmp1,tmp2,left,right)
  top = top or i
  lvl = lvl or 0
  print(fmt("%s%s", string.rep(".",lvl),#i._rows))
  if #i._rows >= 2*(#top._rows)^.5 then
    tmp1, tmp2 = top:half(i._rows)
    if #tmp1._rows < #i._rows then left  = tmp1:cluster(top,lvl+1) end
    if #tmp2._rows < #i._rows then right = tmp2:cluster(top,lvl+1) end
  end
  return {here=i, left=left, right=right} end

function EGS.dist(i,r1,r2)
  local d,n,inc = 0, (#i.x)+1E-31
  for _,col in pairs(i.x) do
    inc = col:dist(r1[col.at], r2[col.at])
    d   = d + inc^2 end
  return (d/n)^.5 end

function EGS.far(i,r1,rows,         fun,tmp)
  fun = function(r2) return {r2, i:dist(r1,r2)} end
  tmp = sort(map(rows,fun), seconds)
  return table.unpack(tmp[#tmp*.9//1] ) end

function EGS.half(i,rows)
  local some,left,right,c,cosine,lefts,rights
  some    = #rows > 512 and many(rows,512) or rows
  left    = i:far(any(rows), some)
  right,c = i:far(left,      some)
  function cosine(r,     a,b)
    a, b = i:dist(r,left), i:dist(r,right); return {(a^2+c^2-b^2)/(2*c),r} end
  lefts,rights = i:clone(), i:clone()
  for n,pair in pairs(sort(map(rows,cosine), firsts)) do
    (n <= #rows/2 and lefts or rights):add( pair[2] ) end
  return lefts,rights,left,right,c end
-- ------------------------------------------------------------------
function any(t) return t[math.random(#t)] end

function bsearch(t,x,      lo,hi,         mid)
  lo,hi = lo or 1,hi or #t
  while lo <= hi do
    io.write(".")
    mid = (lo + hi)//2
    if t[mid] >= x then hi= mid - 1 else lo= mid + 1 end end
  return lo>#t and #t or lo end

function cli(slot,x)
  for n,flag in ipairs(arg) do
    if   flag:sub(1,1)=="-" and slot:match("^"..flag:sub(2)..".*")
    then x= x=="false" and "true" or x=="true" and "false" or arg[n+1] end end
  return thing(x) end

function firsts(a,b)  return a[1] < b[1] end

fmt = string.format

function many(t,n, u) u={};for j=1,n do t[1+#t]=any(t) end; return u end

function map(t,f,        p,u)
  f,u = f or same, {}
  p = debug.getinfo(f).nparams -- only available in  LUA 5.2+
  f= function(k,v) if p==2 then return f(k,v) else return f(v) end end
  for k,v in pairs(t) do push(u, f(k,v)) end; return u end

function new(k,t) k.__index=k; return setmetatable(t,k) end

function o(t,    u)
  if type(t)~="table" then return tostring(t) end
  local key=function(k) return string.format(":%s %s",k,o(t[k])) end
  u = #t>0 and map(t,o) or map(sort(slots(t)),key)
  return '{'..table.concat(u," ")..'}' end

function push(t,x) table.insert(t,x); return x end

function rows(file,       x)
  file = io.input(file)
  return function()
    x=io.read(); if x then return things(x) else io.close(file) end end end

function same(x) return x end

function seconds(a,b) return a[2] < b[2] end

function slots(t, u)
  u={}
  for k,_ in pairs(t) do k=tostring(k); if k:sub(1,1)~="_" then u[1+#u]=k end en
d
  return u end

function sort(t,f)   table.sort(t,f); return t 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(x,sep,   t)
  t={};for y in x:gmatch(sep or"([^,]+)") do push(t,thing(y)) end; return t end

-- ------------------------------------------------------------------
--for row in rows("../../data/auto93.csv") do print(o(row)) end
--local n,i=0,EGS:new()
--for row in rows("../../data/auto93.csv") do n=n+1; i:add(row)  end
--i:cluster()

function our.go.aa() print(1) end

our.help:gsub("\n [-](%S+)[^\n]*%s([^%s]+)",function(k,v) your[k]=cli(k,v) end)

if your.help then print(out.help) end

for _,k in pairs(sort(slots(go))) do
  if
for k,v in pairs(_ENV) do if not b4[k] then print("?",k,type(v)) end end
```