```lua
local the, help = {}, [[
lua tiny.lua [OPTIONS]

A small sample multi-objective optimizer
(c)2021 Tim Menzies <timm@ieee.org> unlicense.org

OPTIONS:
  -best      X   best examples are in 1..best*size(all)    = .05
  -file      X   where to read data                        = ../../data/auto93.csv
  -h             show help                                 = false
  -seed      X   random number seed;                       = 10019
  -stop      X   create subtrees while at least 2*stop egs =  4
  -tiny      X   min range size = size(egs)^tiny           = .5
  -trivial   X   ignore differences under trivial*sddev    = .35 ]]

local b4={};for k,v in pairs(_ENV) do b4[k]=k end

---------------------------------------------------------
local say,fmt,csv,map,copy,mode,norm,push,sort,firsts
fmt = string.format
function say(...)      print(fmt(...)) end
function firsts(x,y)   return x[1] < y[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 copy(t,  u)   u={}; for k,v in pairs(t) do u[k]=v end; return u end
function norm(lo,hi,x) return math.abs(lo-hi)<1E-32 and 0 or (x-lo)/(hi-lo) end
function map(t,f,u)    u={};for k,v in pairs(t) do push(u,f(k,v)) end; return u end

function csv(file,   x)
  file = io.input(file)
  x    = io.read()
  return function(   t,tmp)
    if x then
      t={}
      for y in x:gsub("[\t ]*",""):gmatch"([^,]+)" do push(t,tonumber(y) or y) end
      x = io.read()
      if #t>0 then return t end
    else io.close(file) end end end

local shout,out
function shout(x) print(out(x)) end
function out(t,     u,keys,key1,val1)
  function keys(t,u)
    u={}; for k,_ in pairs(t) do u[1+#u]=k end; return sort(u); end
  function key1(_,k)  return string.format(":%s %s", k, out(t[k])) end
  function val1(_,v)  return out(v) end
  if type(t) ~= "table" then return tostring(t) end
  u = #t>0 and map(t, val1) or map(keys(t), key1)
  return "{"..table.concat(u," ").."}" end

local Seed,randi,rand,ent
Seed = the.seed
function randi(lo,hi) return math.floor(0.5 + rand(lo,hi)) end
function rand(lo,hi)
  lo, hi = lo or 0, hi or 1
  Seed = (16807 * Seed) % 2147483647
  return lo + (hi-lo) * Seed / 2147483647 end

local ent,mode
function ent(t,    n,e)
  n=0; for _,n1 in pairs(t) do n = n + n1 end
  e=0; for _,n1 in pairs(t) do e = e - n1/n*math.log(n1/n,2) end
  return e,n  end

function mode(t,     most,out)
  most = 0
  for x,n in pairs(t) do if n > most then most,out = n,x end end
  return out end

--------------------------
local slurp,sample,ordered,clone
function slurp(out)
  for eg in csv(the.file) do out=sample(eg,out) end
  return out end

function clone(i, inits,     out)
  out = sample(i.heads)
  for _,eg in pairs(inits or {}) do out = sample(eg,out) end
  return out end

function sample(eg,i)
  local numeric,independent,dependent,head,data,datum
  i = i or {n=0,xs={},nys=0,ys={},lo={},hi={},w={},egs={},heads={},divs={}}
  function head(n,x)
    function numeric()     i.lo[n]= math.huge; i.hi[n]= -i.lo[n] end
    function independent() i.xs[n]= x end
    function dependent()
      i.w[n]  = x:find"-" and -1 or 1
      i.ys[n] = x
      i.nys   = i.nys+1 end
    if not x:find":" then
      if x:match"^[A-Z]" then numeric() end
      if x:find"-" or x:find"+" then dependent() else independent() end end
    return x end
  function data(eg) return {raw=eg, cooked=copy(eg)} end
  function datum(n,x)
    if x ~= "?" then
      if i.lo[n] then
        i.lo[n] = math.min(i.lo[n],x)
        i.hi[n] = math.max(i.hi[n],x) end end
    return x end
  eg = eg.raw and eg.raw or eg
  if #i.heads==0 then i.heads=map(eg,head) else push(i.egs,data(map(eg,datum))) end
  i.n = i.n + 1
  return i end

function ordered(i)
  local function better(eg1,eg2,     a,b,s1,s2)
    s1,s2=0,0
    for n,_ in pairs(i.ys) do
      a  = norm(i.lo[n], i.hi[n], eg1.raw[n])
      b  = norm(i.lo[n], i.hi[n], eg2.raw[n])
      s1 = s1 - 2.71828^(i.w[n] * (a-b)/i.nys)
      s2 = s2 - 2.71828^(i.w[n] * (b-a)/i.nys) end
    return s1/i.nys < s2/i.nys end
  for j,eg in pairs(sort(i.egs,better)) do
    if j < the.best*#i.egs then eg.klass="best" else eg.klass="rest" end end
  return i end

```

```lua
local discretize, xys_sd, bin, div
function bin(z,divs)
  if z=="?" then return "?" end
  for n,x in pairs(divs) do
    if x.lo<= z and z<= x.hi then return string.char(96+n) end end end

function discretize(i)
  for col,_ in pairs(i.xs) do
    if i.lo[col] then
      local xys,sd = xys_sd(col, i.egs)
      i.divs[col]  = div(xys, the.tiny*#xys, the.trivial*sd)
      for _,eg in pairs(i.egs) do
        eg.cooked[col]= bin(eg.raw[col], i.divs[col]) end end end
  return i end

function xys_sd(col,egs,    xys,p)
  xys={}
  for _,eg in pairs(egs) do
    local x=eg.raw[col]
    if x~="?" then push(xys, {x=x,  y=eg.klass}) end end
  xys = sort(xys, function(a,b) return a.x < b.x end)
  p   = function(z) return xys[z*#xys//10].x end
  return xys, math.abs(p(.9) - p(.1))/2.56 end

function div(xys,tiny,trivial,     one,all,merged,merge)
  function merged(a,b,an,bn,      c)
    c={}
    for x,v in pairs(a) do c[x] = v end
    for x,v in pairs(b) do c[x] = v + (c[x] or 0) end
    if ent(c)*.99 <= (an*ent(a) + bn*ent(b))/(an+bn) then return c end
  end ------------------------
  function merge(b4)
    local j,tmp = 0,{}
    while j < #b4 do
      j = j + 1
      local now, after = b4[j], b4[j+1]
      if after then
        local simpler = merged(now.has,after.has, now.n,after.n)
        if simpler then
          now = {lo=now.lo, hi=after.hi, n=now.n+after.n, has=simpler}
          j = j + 1 end end
      push(tmp,now) end
    return #tmp==#b4 and b4 or merge(tmp) -- recurse until nothing merged
  end ------------------------
  one = {lo=xys[1].x, hi=xys[1].x, n=0, has={}}
  all = {one}
  for j,xy in pairs(xys) do
    local x,y = xy.x, xy.y
    if   j< #xys-tiny and x~= xys[j+1].x and one.n> tiny and one.hi-one.lo>trivial
    then one = push(all, {lo=one.hi, hi=x, n=0, has={}})
    end
    one.n  = 1 + one.n
    one.hi = x
    one.has[y] = 1 + (one.has[y] or 0); end
  return merge(all) end

local splitter,worth,tree
function splitter(xs, egs)
  function worth(at,_)
    local xy = {}
    for _,eg in pairs(egs) do
      local x,y = eg.cooked[at], eg.klass
      if x ~= "?" then
        xy[x] = xy[x] or {}
        xy[x][y] = 1 + (xy[x][y] or 0) end end
    local n, xpect = #egs, 0
    for _,t in pairs(xy) do
     local e,n1 = ent(t)
     xpect = xpect+n1/n*e end
    return {xpect,at} end
  return sort(map(xs, worth),firsts)[1][2] end

function tree(xs, egs)
  if #egs > 2*the.stop then
    local kid,kids,at = {},0,splitter(xs,egs)
    for _,eg in pairs(egs) do
      local x= eg.cooked[at]
      if not kid[x] then kid[x]={}; kids=kids+1 end
      push(kid[x], eg) end
    if kids> 1 then
      return map(kid, function(x,t) return {at=at,val=x,sub=tree(xs,t)} end) end end
  local t = {}
  for _,eg in pairs(egs) do t[eg.klass] = 1 + (t[eg.klass] or 0) end
  return {mode=mode(t),  has=t} end --egs=egs} end

-- function show(tree,pre)
--   pre = pre or ""
--   if tree.sub then
--     say("%s %s ",pre)
--     for _,one in  pairs(tree.sub) do
--       say("%s %s=%s", pre, one.at or "", one.val or "")
--       show(one.sub,pre.."|.. ") end end
--   else x end end
--

local function main(s)
  local s=discretize(ordered(slurp()))
  for col,divs in pairs(s.divs) do
     print("")
     for _,div in pairs(divs) do
       print(col,out(div)) end end end

------------------------------------------------------------------------------
-- Make 'the' options array from help string and any updates from command line.
(help or ""):gsub("^.*OPTIONS:",""):gsub("\n%s*-([^%s]+)[^\n]*%s([^%s]+)",
   function(flag,x)
     for n,word in ipairs(arg) do if word==("-"..flag) then
       x = x=="false" and "true" or tonumber(arg[n+1]) or arg[n+1] end end
     if x=="false" then x=false elseif x=="true" then x=true end
     the[flag]=x end)

Seed=the.seed or 10019
if the.h then print(help) else main() end
for k,v in pairs(_ENV) do if not b4[k] then print("?rogue: ",k,type(v)) end end
```