```lua
local help=[[
LOOK.LUA: landscape analysis
(c) 2022 Tim Menzies, timm@ieee.org, BSD-2 license
"I think the highest and lowest points are the important ones.
 Anything else is just... in between." ~Jim Morrison

INSTALL: requires: lua 5.4+
         download: lib.lua, look.lua, looked.lua
         test    : lua egs.lua -h

USAGE: lua looked.lua [OPTIONS]
                                       defaults
                                       --------
  --also     -a  size of rest=best*also = 4
  --p        -p  distance coefficient   = 2
  --far      -f  far                    = .95
  --Some     -S  sample size            = 256
  --seed     -s  random number seed     = 10019
  --min      -m  min size pass1         = .5
  --Min      -M  min size pass2         = 10

  --file     -f  csv file with data     = ../../etc/data/auto93.csv
  --help     -h  show help              = false
  --verbose  -v  verbose mode           = false
  --go       -g  start up action        = nothing]]

local _ = require"lib"
local big,is,push,shuffle,tothing = _.big, _.is, _.push, _.shuffle, _.tothing
local the={}
help:gsub(" [-][-]([^%s]+)[^\n]*%s([^%s]+)",function(k,x) the[k]=_.tothing(x)end)
--------------------------------------------------------------------------------
local ROW=is"ROW"
function ROW.new(i,of,cells) i.cells, i.of, i.evaluated = cells,of,0 end
function ROW.better(i,j,        n,s1,s2,v1,v2)
  n,s1,s2 = 0,0,0
  for _,__ in pairs(i.of.ys) do n = n + 1 end
  for c,w in pairs(i.of.ys) do
    v1,v2 = i.of.norm(c, r1[c]), i.of.norm(c, r2[c])
    s1    = s1 - 2.7183^(w * (v1 - v2) / n)
    s2    = s2 - 2.7183^(w * (v2 - v1) / n) end
  return s1/n < s2/n end

function ROW.dist(i,j,     d,n,dist1)
  function dist1(c,v1,v2)
    if v1=="?" and v2=="?" then return 0 end
    if not i.of.nums[c]
    then return v1==v2 and 0 or 1
    else if     v1=="?" then v2=i.of.norm(c,v2); v1= v2<.5 and 1 or 0
         elseif v2=="?" then v1=i.of.norm(c,v1); v2= v1<.5 and 1 or 0
         else   v1,v2 = i.of.norm(c,v1), i.of.norm(c,v2) end
         return math.abs(v1-v2) end
  end ---------------------------
  d,n = 0,0
  for c,_ in pairs(i.xs) do n,d = n+1, d + (dist1(c,i[c], j[c]))^the.p end
  return (d/n)^(1/the.p) end
--------------------------------------------------------------------------------
local ROWS=is"ROWS"
local function num(s)  return s:find"^[A-Z]" end
local function goal(s) return s:find"[!+-]$" end
local function wght(s) return s:find"-$" and -1 or 1 end

function ROWS.new(i,src)
  i.rows, i.nums, i.xs, i.ys, i.names =  {},{},{},{},nil
  if type(src)=="table" then for _,r in pairs(src) do i:add(r) end
                        else for r   in csv(src)   do i:add(r) end end end

function ROWS.add(i,t)
  if   i.names
  then r = t.cells and r or ROW(i,t); i:update(r.cells); push(i.rows, r)
  else i:header(r) end  end

function ROWS.header(i,r)
  i.names = r
  for c,s in pairs(r)do if num(s) then i.nums[c]={lo=big,hi=-big} end end
  for c,s in pairs(r)do if goal(s)then i.ys[c]=wght(s) else i.xs[c]=c end end end

function ROWS.update(i,t,   v)
  for c,num in pairs(i.nums) do
    v = t[c]
    if v ~="?" then num.lo = math.min(v, num.lo)
                    num.hi = math.max(v, num.hi) end end end

function ROWS.norm(i,c,v,   lo,hi)
  lo,hi = i.nums[c].lo, i.nums[c].hi
  return (v=="?" and v) or ((hi-lo) < 1E-9 and 0) or (v-lo)/(hi-lo) end

function ROWS.around(i,r1,t,          fun)
  function fun(r2) return {dist=dist(r1,r2), row=r2} end
  return sort(map(t or i.rows,fun),lt"dist") end

function ROWS.far(i,r1,t,   tmp)
  tmp= i:around(r1,t)
  return tmp[#tmp*the.far//1].row end

function ROWS.look(i,  w,sample,ra,rest)
  w      = shuffle(i.rows)
  sample = many(w, the.Some)
  ra     = i:far(any(sample), sample)
  rest   = {}
  for _,stop in pairs({(#w)^the.min, the.Min})  do
    while #w > stop do
      local rb = far(ra, sample)
      if rb:better(ra) then ra,rb = rb,ra end
      ra.evaluated, rb.evaluated = true,true
      local c = ra:dist(rb)
      for _,rc in pairs(w) do rc.x=(rc:dist(ra)^2 +c^2- rc:dist(rb)^2)/(2*c) end
      local best = {}
      for n,rc in paris(sort(w,lt"x")) do push(n<=#w/2 and best or rest,rc) end
      if #best==#w then break else w=best end
      sample = many(w,the.Some) end end
 return ra,w,many(rest, #w*the.also) end

return {ROWS=ROWS, ROW=ROW, help=help, the=the}


-- i=self
-- j=other
-- k==loop counter
-- v= cell value
-- c = column index
-- r= row (which is just a table)
-- s = string
-- t,u= table

```