```lua
local help=[[
LOOK: landscape analysis
(c) 2022 Tim Menzies, timm@ieee.org, BSD2 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, looking.lua
         test    : lua egs.lua -h

USAGE: lua looking.lua [OPTIONS]

                                          defaults
                                          --------
  --Far   -F  far                         = .95
  --How   -H  how we optimize: more,less,tabu = more
  --Min   -M  min size pass2              = 10
  --Some  -S  sample size                 = 512
  --also  -a  size of rest=best*also      = 4
  --min   -m  min size pass1              = .5
  --p     -p  distance coefficient        = 2
  --seed  -s  random number seed          = 10019

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

local _ = require"lib"
local any,big,csv,is,lt,many,map= _.any, _.big, _.csv, _.is, _.lt, _.many, _.map
local o,oo,per,push,shuffle,sort= _.o, _.oo, _.per, _.push, _.shuffle, _.sort
local tothing                   = _.tothing

local the={}
help:gsub(" [-][-]([^%s]+)[^\n]*%s([^%s]+)",function(k,x) the[k]=_.tothing(x)end)

local function nump(s)  return s:find"^[A-Z].*" end
local function skipp(s) return s:find":$" end
local function goalp(s) return s:find"[!+-]$" end
local function wght(s) return s:find"-$" and -1 or 1 end
local function ranges(col,...)
  local ranges,tmp={},{}
  for klass,rows in pairs{...} do
    for _,row in pairs(rows) do
      local v=row[col.at]
      if v~="?" then
        local bin = col:bin(v)
        tmp[bin]  = tmp[bin] or push(ranges, RANGE(x,x,SYM(col.at, col.txt)))
        tmp[bin]:add(v,klass) end end end
  return col:binsMerge(sort(ranges, lt"lo")) end
-------------------------------------------------------------------------------
local RANGE,ROWS,TREE = is"RANGE", is"ROWS", is"TREE"
local ROW,SYM,NUM     = is"ROW",   is"SYM",  is"NUM"
-------------------------------------------------------------------------------
function ROW.new(i,of,cells) i.cells, i.of, i.evaluated = cells,of,false end
function ROW.__lt(i,j,       n,s1,s2,v1,v2)
  s1, s2, n = 0, 0, #i.of.ys
  for _,col in pairs(i.of.ys) do
    v1,v2 = col:norm(i.cells[col.at]), col:norm(j.cells[col.at])
    s1    = s1 - 2.7183^(col.w * (v1 - v2) / n)
    s2    = s2 - 2.7183^(col.w * (v2 - v1) / n) end
  return s1/n < s2/n end

function ROW.dist(i,j,     d,n)
  d,n = 0,0
  for _,col in pairs(i.of.xs) do
    n = n+1
    d =d + (col:dist(i.cells[col.at], j.cells[col.at]))^the.p end
  return (d/n)^(1/the.p) end
-------------------------------------------------------------------------------
function SYM.new(i,at,txt)
  i.at=at or 0; i.txt=txt or ""; i.all, i.n, i.most, i.mode = {},0,0,nil end

function SYM.dist(i,v1,v2)
  return (v1=="?" and v2=="?" and 1) or (v1==v2 and 0 or 1) end

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

function SYM.div(i,   e)
  e=0; for k,n in pairs(i.all) do e=e-n/i.n*math.log(n/i.n,2) end ;return e end

function SYM.mid(i) return i.mode end
function SYM.bin(i,x) return x end
function SYM.binsMerge(i,ranges) return ranges end

function SYM.merged(i,j,min,  k)
  k = SYM(i.at,i.txt)
  for v,n in pairs(i.all) do k:add(v,n) end
  for v,n in pairs(j.all) do k:add(v,n) end
  min = min or 0
  if i.n < min or j.n < min or k:div() < (i.n*i:div() + j.n*j:div())/k.n then
    return k end end
-------------------------------------------------------------------------------
function RANGE.new(i,lo,hi,y) i.lo,i.hi,i.y = lo, hi, y end

function RANGE.__tostring(i)
  local x, lo, hi = i.y.txt, i.x.lo, i.x.hi
  if     lo ==  hi  then return fmt("%s == %s",x, lo)
  elseif hi ==  big then return fmt("%s >= %s",x, lo)
  elseif lo == -big then return fmt("%s < %s", x, hi)
  else                   return fmt("%s <= %s < %s",lo,x,hi) end end

function RANGE.add(i,v,y)
  if v=="?" then return v else i.lo = math.min(i.lo, v)
                               i.hi = math.max(i.hi, v)
                               i.y:add(y) end end

function RANGE.selects(i,t,     v)
  v = t.cells[i.at]
  return v=="?" or (i.lo==i.hi and i.lo==v) or (i.lo<=v and v<i.hi) end

function RANGE.score(i,goal,B,R, how,    b,r,z)
  how={}
  how.more= function(b,r) return ((b<r or b+r < .05) and 0) or b^2/(b+r) end
  how.less= function(b,r) return ((r<b or b+r < .05) and 0) or r^2/(b+r) end
  how.tabu= function(b,r) return 1/(b+r) end
  b, r, z = 0, 0, 1/big
  for v,n in pairs(i.y.all) do
    if v==goal then b = b+n else r=r+n end end
  return how[the.How or "good"](b/(B+z), r/(R+z)) end
-------------------------------------------------------------------------------
function NUM.new(i,at,txt)
  i.at=at or 0; i.txt=txt or ""; i.w = wght(i.txt)
  i.all,i.n,i.ok,i.lo,i.hi={},0,true,big,-big end

function NUM.add(i,v)
  if v ~="?" then
    i.lo=math.min(v,i.lo);i.hi=math.max(v,i.hi);push(i.all,v); i.ok=false end end

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

function NUM.dist(i,v1,v2)
  if     v1=="?" and v2=="?" then return 0 end
  if     v1=="?"             then v2=i:norm(v2); v1= v2<.5 and 1 or 0
  elseif v2=="?"             then v1=i:norm(v1); v2= v1<.5 and 1 or 0
  else   v1, v2 = i:norm(v1), i:norm(v2) end
  return math.abs(v1-v2) end

function NUM.has(i) if not i.ok then sort(i.all) end;i.ok=true; return i.all end
function NUM.mid(i) return per(i:has(),.5) end
function NUM.div(i,  a) a=i.has(); return (per(a,.9) - per(a,.1))/2.56 end
function NUM.bin(i,x,  b) b=(i.hi-i.lo)/the.bins;return math.floor(x/b+0.5)*b end
function NUM.binsMerge(i,ranges,min,      a,b,c,j,n,tmp,expand)
  function expand(t)
    if #t<2 then return {} end
    for j=2,#t do t[j].lo=t[j-1].hi end
    t[1].x.lo, t[#t].x.hi= -big,big
    return t
  end ------------------
  j,n,tmp = 1,#ranges,{}
  while j<=n do
    a, b = ranges[j], ranges[j+1]
    if b then
      c = a.y:merge(b.y,min)
      if c then a = {lo=a.lo, hi=b.hi, y=c}
           j = j+1 end end
    tmp[#tmp+1] = a
    j = j+1 end
  return #tmp==#ranges and expand(tmp) or i:binsMerge(tmp,min) end
-------------------------------------------------------------------------------
function ROWS.new(i,src)
  i.all, i.cols, 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.clone(i,inits,    j)
  j=ROWS((i.names)); for _,r in pairs(inits or {}) do j:add(r) end; return j end

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

function ROWS.header(i,t,      col)
  i.names = t
  for at,txt in pairs(t) do
    col = push(i.cols, (nump(txt) and NUM or SYM)(at,txt))
    if not skipp(txt) then push(goalp(txt) and i.ys or i.xs, col) end end end

function ROWS.update(i,t)
  for _,col in pairs(i.cols) do col:add(t[col.at]) end end

function ROWS.around(i,r1,t,           fun)
  function fun(r2) return {dist=r1:dist(r2), row=r2} end
  return sort(map(t or i.all, 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.mid(i,cols) return map(cols or i.ys, function(col) return col:mid() end) end
function ROWS.lo(i,cols) return map(cols or i.ys, function(col) return col.lo end) end

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

function ROWS.how(i, bests, rests)
  local bins={}
  for _,col in pairs(XXX.xs) do
    for _,bin in pairs(ranges(col, bests, rests)) do
      push(bins,{score=bin:score(1,#bests,#rests), bin=bin}) end end
  for _,bin in pairs(sort(bins,gt"score")) do print(bin) end end

-------------------------------------------------------------------------------
return {NUM=NUM,ROWS=ROWS, ROW=ROW, help=help, the=the}
```