```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
                                      --------
  --also  -a  size of rest=best*also  = 4
  --p     -p  distance coefficient    = 2
  --Far   -F  far                     = .95
  --Some  -S  sample size             = 512
  --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
  --loud  -l  verbose mode            = false
  --go    -g  start up action         = nothing]]

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 ROW,ROWS,SYM,NUM = is"ROW", is"ROWS", 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 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,1E32,-1E32 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 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
----------------------------------------------------------------------------
return {NUM=NUM,ROWS=ROWS, ROW=ROW, help=help, the=the}

```

```lua
 _    _ ___    _    _  _ ____
|    | |__]   |    |  | |__|
|___ | |__] .|___ |__| |  |

-- vim: ts=2 sw=2 et :
-- LIB.LUA : misc support code.
-- (c) 2022 Tim Menzies.  BSD-2 license.
local b4={}; for k,_ in pairs(_ENV) do b4[k]=k end
local fmt =string.format
local rand=math.random
local big = 1E32

local function any(t)       return t[math.random(#t)] end
local function many(t,n, u) u={};for j=1,n do u[1+#u]=any(t) end; return u end
local function lt(x)        return function(a,b) return a[x]<b[x] end end
local function push(t,x)    t[1+#t]=x; return x end
local function sort(t,f)    table.sort(t,f); return t end
local function map(t,f,  u) u={}; for k,v in pairs(t) do u[1+#u]=f(v) end
                            return u end

local function per(t,p, i) i=p*#t//1; return t[math.max(1,math.min(#t,i))] end

local function shuffle(t,    j)
  for i = #t, 2, -1 do j=rand(i); t[i],t[j] = t[j],t[i] end
  return t end

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

local function csv(csvfile)
  csvfile = io.input(csvfile)
  return function(line, t)
    line=io.read()
    if not line then io.close(csvfile) else
      t={}; for x in line:gmatch("([^,]+)") do t[1+#t]=tothing(x) end
      return t end end end

local function cli(d,help)
  d = d or {}
  for key,x in pairs(d) do
    x = tostring(x)
    for n,flag in ipairs(arg) do
      if flag==("-"..key:sub(1,1)) or flag==("--"..key) then
        x = x=="false" and"true" or x=="true" and"false" or arg[n+1] end end
    d[key] = tothing(x) end
  if d.help then return os.exit(print(
     help:gsub("[%u][%u%d]+", "\27[31m%1\27[0m")
         :gsub("\"[^\"]+\"",  "\27[32m%1\27[0m")
         :gsub("(%s)([-][-]?[^%s]+)(%s)","%1\27[33m%2\27[0m%3"),"")) end
  return d end

local function o(t,    u)
  if #t>0 then return "{"..table.concat(map(t,tostring)," ").."}" end
  u={}; for k,v in pairs(t) do u[1+#u] = fmt(":%s %s",k,v) end
  return (t.is or "").."{"..table.concat(sort(u)," ").."}" end

local function oo(x) print(o(x)) end

local function is(name,    t,new)
  function new(kl,...) local x=setmetatable({},kl); kl.new(x,...); return x end
  t = {__tostring=o, is=name or ""}; t.__index=t
  return setmetatable(t, {__call=new}) end

local function main(funs,settings)
  local defaults, names, fails = {}, {}, 0
  for k,f in pairs(funs) do
    if type(f)=="function" then push(names,k) end end
  for k,v in pairs(settings) do
    defaults[k]=v end
  if funs[settings.go] then
    names={settings.go} end
  for _,one in pairs(sort(names))  do         -- for all we want to do
    for k,v in pairs(defaults) do
      settings[k]=v end                      -- reset the settings to defaults
    math.randomseed(settings.seed or 10019)  -- reset random number seed
    io.stderr:write(".")
    local status = funs[one]()               -- run demo
    if status ~= true then
      print("-- Error",one,status)
      fails = fails + 1 end end              -- update fails
  for k,v in pairs(_ENV) do if not b4[k] then print("?",k,type(v)) end end
  os.exit(fails) end

return {any=any, big=big, cli=cli, csv=csv, fmt=fmt, is=is, lt=lt, oo=oo, o=o,
        main=main, many=many, map=map, per=per, push=push, rand=rand,
        shuffle=shuffle, sort=sort, tothing=tothing}

 _    ____ ____ _  _ _ _  _ ____    _    _  _ ____
 |    |  | |  | |_/  | |\ | | __    |    |  | |__|
 |___ |__| |__| | \_ | | \| |__] .  |___ |__| |  |

-- vim: ts=2 sw=2 et :
-- LOOK.LUA: landscape analysis
-- (c) 2022 Tim Menzies, timm@ieee.org, BSD-2 license
local l,L  = require"lib", require"look"
local any,cli,csv,fmt       =  l.any, l.cli, l.csv, l.fmt
local lt, main, many, map  = l.lt, l.main, l.many,l.map
local o, oo,per,shuffle,sort   = l.o, l.oo, l.per, l.shuffle, l.sort
local NUM,ROW,ROWS             = L.NUM, L.ROW, L.ROWS
local the                  = cli(L.the,L.help)
--------------------------------------------------------------------------------
local go,no={},{} -- place to store enabled and disabled tests

function go.the()
  if the.loud then oo(the) end; return type(the.seed)=="number" end

function go.row(    n)
  n=0
  for r in csv(the.file) do n=n+#r; if the.loud then oo(r) end end
  return n == 3192 end

function go.egs(    rows)
  rows= ROWS(the.file)
  if the.loud then map(rows.ys,oo) end
  return rows.ys[1].hi==5140 and rows.ys[1].lo==1613 end

function go.clone(    rows)
  rows= ROWS(the.file)
  return rows:mid()[3]==20 end

function go.dist(    r1,rows,ok)
  ok,rows= true, ROWS(the.file);
  r1 = rows.all[1]
  for _,r2 in pairs(rows.all) do
    ok = ok and r2:dist(r2)==0
    ok = ok and r1:dist(r2) == r2:dist(r1) end
  return ok end

function go.around(    r1,rows, order)
  rows = ROWS(the.file);
  r1 = rows.all[1]
  order = rows:around(r1)
  return order[#order//3].dist < order[#order//2].dist  end

function go.far(    rows,r1,r2,ok)
  ok = true
  rows = ROWS(the.file);
  for k=1,10 do
    r1 = rows:far(any(rows.all))
    r2 = rows:far(r1)
    ok = ok and r1:dist(r2) > .5 end
  return ok end

function go.betters(  t,n1)
  t=sort(ROWS(the.file).all)
  n1=10
  for k =1,n1 do oo(t[k].cells) end; print""
  for k =#t-n1, #t do oo(t[k].cells) end
  return t[1] < t[#t]
end

function go.look(   rows,best,bests,rests,n,names,b4,guess,b,g)
  rows = ROWS(the.file)
  names=map(rows.ys,function(col) return col.txt end)
  b=NUM()
  g=NUM()
  b4=rows:mid()
  for i=1,10 do
    rows = ROWS(the.file)
    rows.all = shuffle(rows.all)
    best,bests,rests = rows:look()
    for n,r in pairs(sort(rows.all)) do r.rank = math.floor(100*n/#rows.all //1) end
    n=0;for _,r in pairs(rows.all) do if r.evaluated then n=n+1 end end
    guess=rows:clone(many(rows.all,n))
    for _,rank in pairs(map(sort(bests,lt"rank"),function(r) return r.rank end)) do b:add(rank) end
    for _,rank in pairs(map(sort(guess.all,lt"rank"),function(r) return r.rank end)) do g:add(rank) end
    print(fmt("%20s %20s %20s",
          o(names),o(b4),
          o(rows:clone(bests):mid()),
          o{bests=#bests,rests=#rests,evalled=n})) end
  for _,p in pairs{0,.2,.4,.6,.8}  do io.write(per(b:has(),p)," ") end; print""
  for _,p in pairs{0,.2,.4,.6,.8}  do io.write(per(g:has(),p)," ") end; print""
  return true end
--------------------------------------------------------------------------------
main(go, the)
```