look.lua

```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             = 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
  --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,push,shuffle,sort     = _.o, _.oo, _.push, _.shuffle, _.sort
local tothing                    = _.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,false end
function ROW.__lt(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, i.cells[c]), i.of:norm(c, j.cells[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.of.xs) do
    n,d = n+1, d + (dist1(c,i.cells[c], j.cells[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.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 r or ROW(i,t); i:update(r.cells); push(i.rows, r)
  else i:header(t) end end

function ROWS.header(i,t)
  i.names = t
  for c,s in pairs(t) do if num(s) then i.nums[c]={lo=big,hi=-big} end end
  for c,s in pairs(t) 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,rl,t,          fun)
  function fun(r2) return {dist=rl:dist(r2), row=r2} end
  return sort(map(t or i.rows, fun), lt"dist") end

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

function ROWS.mid(i,cols)
  local function mid(c,t)
    if i.nums[c] then
      local s,n,v = 0,0
      for _,r in pairs(i.rows) do v=r[c]; if v~="?" then n=n+1;s=s+r[c] end end
      return s/n
    else
      local most,mode,tmp,v = 0,nil,{}
      for _,r in pairs(i.rows) do
        v=r[c]; if v~="?" then tmp[v] = 1 + (tmp[v] or 0) end end
      for x,n in pairs(tmp) do if n>most then mode,most = x,n end end
      return mode end
   end --------------
   out={}; for c,_ in pairs(cols or i.ys) do out[c] = mid(c,i.rows) end
   return out end

function ROWS.look(i,  w,sample,best,rests)
  w      = i.rows
  sample = many(w, the.Some)
  best   = i:far(any(sample), sample)
  rests  = {}
  for _,stop in pairs({2*(#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
      if #bests==#w then break else w=bests end
      sample = many(w,the.Some) end end
 return ra,w,many(rests, #w*the.also) end

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

lib.lua

```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 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, 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,main,map = l.any, l.cli, l.csv, l.main, l.map
local o, oo,shuffle,sort   = l.o, l.oo, l.shuffle, l.sort
local ROW,ROWS             = 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.nums,oo) end
  return rows.nums[1].hi==8 end

function go.clone(    rows)
  rows= ROWS(the.file)
  oo(rows:mid()) end

function go.dist(    r1,rows,ok)
  ok,rows= true, ROWS(the.file);
  r1 = rows.rows[1]
  for _,r2 in pairs(rows.rows) 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.rows[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=1,50 do
    r1 = rows:far(any(rows.rows))
    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).rows)
  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(   rs,best,bests,rests,n)
  for i=1,20 do
    print("")
    rs = ROWS(the.file)
    rs.rows = shuffle(rows.rows)
    best,bests,rests = rs:look()
    for n,r in pairs(sort(rs.rows)) do r.rank = n // (#rows.rows // (6/.35)) end
    for _,r in pairs(bests) do print(r.rank) end
    n=0
    for _,r in pairs(rs.rows) do if r.evaluated then n=n+1 end end
    oo{bests=#bests,rests=#rests,n=n} end
  return true end
--------------------------------------------------------------------------------
main(go, the)
```