```lua
#!/usr/bin/env lua
local l=require("lib")
local the=l.settings[[

L5 : a lean little learning library, in LUA
(c) 2022 Tim Menzies <timm@ieee.org> BSD-2 license

USAGE: lua l5.lua [OPTIONS]

OPTIONS:
 -e  --eg       start-up example                       = nothing
 -b  --bins     max number of bins                     = 8
 -d  --dump     on test failure, exit with stack dump  = false
 -f  --file     file with csv data                     = ../data/auto93.csv
 -F  --Far      how far to look for poles (max=1)      = .95
 -h  --help     show help                              = false
 -m  --min      min size. If<1 then t^min else min.    = 10
 -n  --nums     number of nums to keep                 = 512
 -p  --p        distance calculation coefficient       = 2
 -r  --rest     size of "rest" set                     = 3
 -s  --seed     random number seed                     = 10019
 -S  --Sample   how many numbers to keep               = 10000 ]]

local any,copy,csv,lt,many,map = l.any,l.copy,l.csv,l.lt,l.many,l.map
local o,obj,oo,per,push     = l.o,l.obj,l.oo,l.per, l.push
local rnd,sort              = l.rnd, l.sort
local Data,Num,Row,Skip,Sym

-- ------------------------------------------------------------------------
Skip=obj"Skip"
function Skip:new(c,x) return {at=c,txt=x} end
function Skip:add(x)   return x end
function Skip:dist(v1,v2) return 0,0 end

-- ------------------------------------------------------------------------
Sym=obj"Sym"
function Sym:new(c,x) return {at=c or 0,txt=x or "",has={}} end
function Sym:add(x)   if x~="?" then self.has[x]=1+(self.has[x] or 0) end end
function Sym:discretize(x) return x end
function Sym:dist(v1,v2)
  return  (v1=="?" and v2=="?" and 1 or v1==v2 and 0 or 1),1 end

-- ------------------------------------------------------------------------
Num=obj"Num"
function Num:new(c,x)
  return {at=c or 0,txt=x or "",lo=1E32,hi=-1E32, has={},
          w=(x or ""):find"-$" and -1 or 1} end
function Num:add(x)
  if x~="?" then self.lo = math.min(x,self.lo)
                 self.hi = math.max(x,self.hi)
                 push(self.has,x) end end
function Num:discretize(x,    tmp)
  tmp = (self.hi - self.lo)/(the.bins - 1)
  return self.lo == self.hi and 1 or math.floor(x/tmp+.5)*tmp end

function Num:norm(n)
  return n=="?" and n or (n-self.lo)/(self.hi-self.lo + 1E-32) end
function Num:pers(t,    a)
  a=sort(self.has)
  return map(t,function(p) return per(a,p) end) end

function Num:dist(v1,v2)
  if   v1=="?" and v2=="?" then return 1,1 end
  v1,v2 = self:norm(v1), self:norm(v2)
  if v1=="?" then v1 = v2<.5 and 1 or 0 end
  if v2=="?" then v2 = v1<.5 and 1 or 0 end
  return math.abs(v1-v2),1 end

-- ------------------------------------------------------------------------
Row=obj"Row"
function Row:new(data,t) return {_data=data,cells=t} end
function Row:around(rows)
  return sort(map(rows, function(r) return {row=r,d=self-r} end),lt"d") end
function Row:far(rows)
  return per(self:around(rows),the.far).row end

function Row:__sub(row,    d,n,d1,n1)
  d,n = 0,0
  for i,col in pairs(self._data.cols.x) do
    d1,n1= col:dist(self.cells[col.at], row.cells[col.at])
    n = n + n1
    d = d + d1^the.p end
  return (d/n)^(1/the.p) end

function Row:__lt(row)
  self.evaled, row.evaled = true,true
  local s1,s2,d,n,x,y=0,0,0,0
  local ys = self._data.cols.y
  for _,col in pairs(ys) do
    x,y= self.cells[col.at], row.cells[col.at]
    x,y= col:norm(x), col:norm(y)
    s1 = s1 - 2.71828^(col.w * (x-y)/#ys)
    s2 = s2 - 2.71828^(col.w * (y-x)/#ys) end
  return s1/#ys < s2/#ys end


-- ------------------------------------------------------------------------
Data=obj"Data"
function Data:new(src)
  self.rows, self.cols = {}, {all={},x={},y={}}
  if   type(src)=="string"
  then csv(src,       function(row) self:add(row) end)
  else map(src or {}, function(row) self:add(row) end) end end

function Data:add(row,    what)
  function what(x)
    return x:find":$" and Skip or (x:find"^[A-Z]" and Num or Sym) end
  if   #self.cols.all==0
  then for c,x in pairs(row) do
         local col = push(self.cols.all, what(x)(c,x))
         push(x:find"[+-]" and self.cols.y or self.cols.x, col) end
  else row = row.cells and row or Row(self,row)
       for c,col in pairs(self.cols.all) do col:add(row.cells[c]) end
       push(self.rows, row) end end

function Data:cheat()
  for i,row in pairs(sort(self.rows)) do
    row.rank = 1+math.floor(100*i/#self.rows)
    row.evaled = false end
  self.rows = l.shuffle(self.rows) end

function Data:half(rows,  above,     some,x,y,c,rxs,xs,ys)
  rows = rows or self.rows
  some = many(rows, the.Sample)
  x    = above or any(some):far(some)
  y    = x:far(some)
  c    = x - y
  rxs  = function(r) return {r=r,x=((r-x)^2 + c^2 - (r-y)^2)/(2*c)} end
  xs,ys= {},{}
  for j,rx in pairs(sort(map(rows,rxs),lt"x")) do
    push(j<=#rows/2 and xs or ys, rx.r) end
  return {xs=xs, ys=ys, x=x, y=y, c=c} end

function Data:best(rows,  above,stop)
  rows = rows or self.rows
  stop = stop or (the.min >=1 and the.min or (#rows)^the.min)
  if   #rows < stop
  then return rows
  else local node = self:half(rows,above)
       if     node.x < node.y
       then   return self:best(node.xs, node.x, stop)
       else   return self:best(node.ys, node.y, stop) end end end


-- ------------------------------------------------------------------------
local eg = {}
local function run(    fails,old)
  fails=0
  the = l.cli(the)
  old = copy(the)
  for k,fun in pairs(eg) do
    if the.eg == "all" or the.eg == k then
      for k,v in pairs(old) do the[k]=v end
      math.randomseed(the.seed)
      print("\n>>>>>",k)
      if not fun() then fails = fails+1 end end end
  l.rogues()
  os.exit(fails) end

function eg.the() oo(the); return true end

function eg.num(  z)
  z=Num(); for i=1,100 do z:add(i) end; print(z); return true end

function eg.sym(  z)
  z=Sym(); for _,x in pairs{1,1,1,1,2,2,3} do z:add(x) end;
  print(z); return true end

function eg.data( d)
  d=Data(the.file);  map(d.cols.x,print) return true end

function eg.dist(    num,d,r1,r2,r3)
  d=Data(the.file)
  num=Num()
  for i=1,20 do
    r1=l.any(d.rows)
    r2=l.any(d.rows)
    r3=r1:far(d.rows)
    io.write(rnd(r3-r1)," ")
    num:add(rnd(r2-r1)) end
  oo(sort(num.has))
  return true end

function eg.sort(     d)
  d = Data(the.file)
  sort(d.rows)
  for i=1,#d.rows,32 do print(i,o(d.rows[i].cells)) end end

function eg.half(     num)
  num=Num()
  for i=1,20 do
    local d = Data(the.file)
    d:cheat()
    map(d:best(),function(row) num:add(row.rank) end) end
  oo(num:pers{.1,.3,.5,.7,.9})
  return end

run()
```