```lua
-- vim: ts=2 sw=2 et :
-- ego.lua : simple landscape analysis (code that is "conscious" of shape of data)
-- (c) 2022 Tim Menzies.  Usage of the works is permitted provided that this
-- instrument is retained with the works, so that any entity that uses the works
-- is notified of this instrument. DISCLAIMER:THE WORKS ARE WITHOUT WARRANTY.

local help=[[
ego.lua: landscape analysis (being 'conscious' of shape of data)
(c) 2022 Tim Menzies, timm@ieee.org
"Don't you believe what you've seen or you've heard,
 'ego' is not a dirty word" ~ Greg Macainsh

INSTALL:
  requires: lua 5.4+
  download: etc.lua, ego.lua, egs.lua
  test    : lua egs.lua -h

USAGE:
  lua egs.lua [OPTIONS]

OPTIONS:                                    default
                                            -------
  -A  --Also  rest is 'also'*Best           = 3
  -B  --Best  use #t^Best as 'best'         = .5
  -b  --bins  max bins for numeric          = 16
  -G  --Goal  goal;  one of: up,down,over   = up
  -k  --keep  #numerics to keep per column  = 256
  -s  --seed  random number seed            = 10019

OPTIONS (other):
  -f  --file  csv file with data            = ../etc/data/auto93.csv
  -h  --help  show help                     = false
  -g  --go    start up action               = nothing ]]

local etc=require"etc"
local big,cli,csv,fmt         =etc.big, etc.cli, etc.csv, etc.fmt
local is,lt,map,o,o,push      =etc.is,  etc.lt,  etc.map, etc.o, etc.oo,etc.push
local splice,sort,string2thing=etc.splice, etc.sort, etc.string2thing
local the = {}
--------------------------------------------------------------------------------
local SOME,NUM,SYM,ROWS = is"SOME", is"NUM", is"SYM", is"ROWS"

local function merge(ranges,min,       a,b,ab,j,n,tmp)
  if ranges[1].x.is == "SYM" then return ranges end
  j,n,tmp = 1,#ranges,{}
  while j<=n do
    a, b = ranges[j], ranges[j+1]
    if b then
      ab = a.y:clone():inject(a.y,b.y)
      if a.n<min or b.n<min or (
         ab:div() < (a.y:div()*a.y.n + b.y:div()*b.y.n)/ab.n)
      then a = {x=a.x:clone():inject(a.x,b.x),   y=y}
           j = j+1 end end
    tmp[#tmp+1] = a
    j = j+1 end
  if #tmp < 2        then return {} end           -- distribution has no splits
  if #tmp < #ranges then return merge(tmp,min) end
  for j=2,#tmp do tmp[j].x.lo = tmp[j-1].x.hi end -- fill in any gaps
  tmp[1].x.lo, tmp[#tmp].x.hi = -big, big          -- stretch across all numbers
  return tmp end

--------------------------------------------------------------------------------
function SYM.new(i,at,name) i.n,i.txt,i.at,i.has = 0,txt or "",at or 0,{} end
function SYM.add(i,x,inc)
  inc = inc or 1
  if x~="?" then i.n = i.n+inc; i.has[x]= inc+(i.has[x] or 0) end end

function SYM.clone(i) return SYM(i.at,i.txt) end
function SYM.inject(i,...)
  for _,more in pairs{...} do for x,n in pairs(more.has) do i:add(x,n) end end
  return i end

function SYM.div(i, e)
  e=0;for _,v in pairs(i.has) do if n>0 then e=e-v/i.n*math.log(v/i.n,2) end end
  return e end

function SYM.range(i,x) return x end

function SYM.want(u,goal,B,R,how,   b,r,z)
  local how={
    good= function(b,r) return ((b<r or b+r < .05) and 0) or b^2/(b+r) end,
    bad=  function(b,r) return ((r<b or b+r < .05) and 0) or r^2/(b+r) end,
    novel=function(b,r) return 1/(b+r) end}
  b, r, z  = 0, 0, 1/big
  goal = goal~=nil and goal or true
  for x,n in pairs(i.has) do
    if x==goal then b=b+n else r=r+n end end
  return how[the.Goal or "good"](b/(B+z), r/(R+z)) end

function SYM.select(i,t)   x=t[i.at]; return x=="?" or i.has[x] end
--------------------------------------------------------------------------------
function SOME.new(i) i.has, i.ok, i.n = {}, false,0 end
function SOME:all() if not i.ok then sort(i.has) end;i.ok=true; return i.has end
function SOME.add(i,x)
  i.n = 1 + i.n
  if     #i.has < the.keep     then i.ok=false; push(i.has,x)
  elseif rand() < the.keep/i.n then i.ok=false; i.has[rand(#i.has)]=x end end
--------------------------------------------------------------------------------
function NUM.new(i,at,txt)
  i.n,i.mu,i.m2,i.sd,i.txt,i.at = 0,0,0,0,txt or "",at or 0
  i.w,i.lo,i.hi,i.has          = i.txt:find"-$" and -1 or 1,big,-big,SOME() end

function NUM.add(i,x,   d)
  if x~="?" then
    i.has:add(x)
    i.n  = i.n+1
    d    = i.mu - x
    i.mu = i.mu + d/i.n
    i.m2 = i.m2 + d*(x - i.mu)
    i.sd = (i.n<2 or i.m2<0) and 0 or (i.m2/(i.n-1))^0.5
    i.lo = math.min(x, i.lo)
    i.hi = math.max(x, i.hi) end end

function NUM.clone(i) return NUM(i.at,i.txt) end
function NUM.inject(i,...)
  for _,more in pairs{...} do for _,n in pairs(more.has.has) do i:add(n) end end
  return i end

function NUM.div() return i.sd end


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

function NUM.range(i,x,n,   b) b=(i.hi-i.lo)/n; return math.floor(x/b+0.5)*b end
function NUM.select(i,t) x=t[i.at]; return x=="?" or i.lo <= x and x <= i.hi end
--------------------------------------------------------------------------------
function ROWS.new(i, src)
  i.names, i.has, i.cols, i.x, i.y = {}, {}, {}, {}, {}
  if type(src)=="table"
  then for _,row in pairs(src) do i:add(row) end
  else for   row in csv(  src) do i:add(row) end end end

function ROWS.add(i,row)
  if    #i.names > 0
  then  push(i.has,row)
        for _,col in pairs(i.cols) do col:add(row[col.at]) end
  else  i.names = row
        for at,txt in pairs(row) do
          local col = push(i.cols, (txt:find"^[A-Z]" and NUM or SYM)(at,txt))
          if not txt:find":$" then
            if txt:find"!$" then i.klass=col end
            push(txt:find"[!+-]$" and i.y or i.x, col) end end end end

function ROWS.betters(i)
  return sort(i.has, function(r1,r2)
                       local s1,s2,e,y,a,b = 0,0,math.exp(1),i.y
                       for _,col in pairs(y) do
                         a,b = col:norm(r1[col.at]), col:norm(r2[col.at])
                         s1 = s1 - e^(col.w * (a - b) / #y)
                         s2 = s2 - e^(col.w * (b - a) / #y) end
                       return s1/#y < s2/#y end) end

function ROWS.xx1(col,yklass,j,y,seen)
  x=i.has[j][col.at]
  if x~="?" then
    bin= col:range(x)
    seen[bin] = seen[bin] or {x=col:clone(), y=yklass()}
    seen[bin].x:add(x)
    seen[bin].y:add(y) end end

function ROWS.xx(i)
  i.rows = i:betters()
  n = (#i.has)^the.Best
  step = (#i.has - n1)/(the.Also*n1)
  for _,col in pairs(i.x) do
    tmp={}
    for j=1,n,1            do i:xx1(col,SYM,j,true, tmp) end
    for j=n+1,#i.rows,step do i:xx1(col,SYM,j,false,tmp) end end end
--------------------------------------------------------------------------------
return {SOME=SOME,NUM=NUM,SYM=SYM,ROWS=ROWS,help=help}
```

```lua
-- vim: ts=2 sw=2 et :
-- etc.lua : misc support code.
-- (c) 2022 Tim Menzies.  Usage of the works is permitted provided that this
-- instrument is retained with the works, so that any entity that uses the works
-- is notified of this instrument. DISCLAIMER:THE WORKS ARE WITHOUT WARRANTY.

local M={}
M.b4={}; for k,_ in pairs(_ENV) do M.b4[k]=k end

M.big =1E32
M.fmt =string.format
M.rand=math.random

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

function M.settings(help)
  --                                  (--longFlag)
  --                  (-c)             --(slot)               (default)
  local pattern="\n ([-][^%s]+)[%s]+([-][-]([^%s]+))[^\n]*%s([^%s]+)"
  local d={}
  help:gsub(pattern, function(c,longFlag,slot,x)
    for n,flag in ipairs(arg) do
      if flag==c or flag==longFlag then
        x = x=="false" and"true" or x=="true" and"false" or arg[n+1] end end
    d[slot] = M.string2thing(x) end)
  if d.help then
    print(help:gsub("%u%u+", "\27[1;33m%1\27[0m")
              :gsub('^[^\n]+\n',"\27[1;33m%1\27[0m")
              :gsub('"[^"]+"',"\27[1;30m%1\27[0m")
        :gsub("(%s)([-][-]?[^%s]+)(%s)","%1\27[1;30m%2\27[0m%3"),"")
    os.exit(0)
  else return d end end

function M.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 M.push(t,M.string2thing(x)) end
      return t end end end

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

function M.splice(t,i,j,k,    u)
  u={}; for n=(i or 1), (j or #t),(k or 1) do u[1+#u] = t[n] end return u end

function M.string2thing(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

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

return M
```

```lua
-- egs.lua : example usage of the ego.lua
-- (c) 2022 Tim Menzies.  Usage of the works is permitted provided that this
-- instrument is retained with the works, so that any entity that uses the works
-- is notified of this instrument. DISCLAIMER:THE WORKS ARE WITHOUT WARRANTY.
local etc=require"etc"
local ego= require"ego"
local map,o,oo,push,sort = etc.map, etc.o, etc.oo, etc.push, etc.sort
local csv,splice = etc.csv, etc.splice
local the = ego.the
local EGS,ROWS = ego.EGS, ego.ROWS
local go,no={},{} -- place to store enabled and disabled tests

function go.the() return type(the.seed) == "number" end
function go.map() return 100==map({10,20,30},function(x) return x*10 end)[1] end
function go.splice(   t)
  t=splice( { 10,220,230,240,250,260,270,280,290,110,320,330,340,350,360,370,380,390,
             210,420,430,440,450,460,470,480,490,210,520,530,540,550,560,570,580,590},
             10,36,4)
  return t[#t]==570 end

function go.csv(    n)
  n=0; for t in csv("../etc/data/auto93.csv") do
    if n>100 and type(t[1]) ~= "number" then return "bad type" end
    n=n+#t end
  return n==3192 end

function go.egs(    n) ROWS("../etc/data/auto93.csv")  end

--------------------------------------------------------------------------------
local function demos(    fails,names,defaults,status)
  fails=0     -- this code will return number of failures
  names, defaults = {},{}
  for k,f in pairs(go) do if type(f)=="function" then etc.push(names,k) end end
  for k,v in pairs(the) do defaults[k]=v end
  if go[the.go] then names={the.go} end
  for _,one in pairs(sort(names))  do         -- for all we want to do
    for k,v in pairs(defaults) do the[k]=v end -- set settings to defaults
    math.randomseed(the.seed or 10019)         -- reset random number seed
    io.stderr:write(".")
    status = go[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 etc.b4[k] then print("?",k,type(v)) end end
  return fails end                             -- return total failure count

the = etc.settings(ego.help)
os.exit(demos())
```