```
-- vim: ts=2 sw=2 et:
local b4={}; for k,_ in pairs(_ENV) do b4[k]=k end
local help = [[
LESS: best or rest multi-objective optimization.
(c) 2022 Tim Menzies, timm@ieee.org
"I think the highest and lowest points are the important ones.
 Anything else is just...in between." ~ Jim Morrison

USAGE: lua less.lua [OPTIONS]

OPTIONS:
  -b  --bins  max bins                = 16
  -s  --seed  random number seed      = 10019
  -S  --some  number of nums to keep  = 256

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

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 the={}
local big,csv,eg,entropy,fmt,main,map,mid,mode,mu,norm,num,o,oo,per,push
local rand,range,sort,some,same,sd,string2thing,sym,thes
local NUM,SYM,RANGE,EGS,COLS
-------------------------------------------------------------------------------
big=math.huge
rand=math.random
fmt=string.format

function same(x)    return x end
function push(t,x) t[1+#t]=x; return x end
function sort(t,f) table.sort(#t>0 and t or map(t,same), f); return t end
function map(t,f, u)  u={}; for k,v in pairs(t) do u[1+#u]=f(v) end;return u end

function 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 csv(src)
  src = io.input(src)
  return function(line, row)
    line=io.read()
    if not line then io.close(src) else
      row={}; for x in line:gmatch("([^,]+)") do push(row,string2thing(x)) end
      return row end end end

function oo(t) print(o(t)) end
function o(t,    u)
  if #t>0 then return "{"..table.concat(map(t,tostring)," ")..")" else
    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 end
-------------------------------------------------------------------------------
function NUM(at,txt) return {at=0,txt="", lo=big,hi=-big,  nump=true,
                             n=0, mu=0, m2=0, sd=0,
                             w=(txt or""):find"-$" and -1 or 1} end

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

function norm(i,x)
  return i.hi - i.lo < 1E-10 and 0 or (x-i.lo)/ (i.hi - i.lo + 1/big) end

function discretize(i,x,n,  b) b=(i.hi-i.lo)/n; return math.floor(x/b+0.5)*b end

function gap(i, x,y)
  if     x=="?" and y=="?" then return 1 end
  if     x=="?"            then y = norm(i,y); x = y<.5 and 1 or 0
  elseif y=="?"            then x = norm(i,x); y = x<.5 and 1 or 0
  else x,y = norm(i,x), norm(i,y) end
  return math.abs(x - y) end
-------------------------------------------------------------------------------
function SYM(at,txt) return {at=0, txt="", n=0, all={}} end
function sym(i,x,n)
  if x=="?" then return x end
  i.n=i.n+1; i.all[x] = (n or 1) + (i.all[x] or 0) end

function mode(i)
  m=0; for y,n in pairs(i.all) do if n>m then m,x=n,y end end; return x end

function entropy(i,   n,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 merged(i,j,n0,    k)
  k = SYM(i.at, i.txt)
  for x,n in pairs(i.all) do sym(k,x,n) end
  for x,n in pairs(j.all) do sym(k,x,n) end
  if i.n<(n0 or 0) or j.n<(n0 or 0) or
     (entropy(i)*i.n + entropy(j)*j.n)/k.n > entropy(h) then return k end end
-------------------------------------------------------------------------------
function RANGE(col,x) return {col=col, x={lo=x,hi=x}, y=SYM()} end
function range(i,x,y)
  if x=="?" then return x end
  i.lo=math.min(i.lo,x)
  i.hi=math.max(i.hi,x)
  sym(i.y, y) end

function rangeB4(i,j) return i.col.at == j.col.at and i.lo < j.lo end
-------------------------------------------------------------------------------
function ROW(eg, cells) return {context=eg, cells=cells} end

function rowB4(i,j,      s1,s2,e,y,a,b)
  y = i.context.cols.y
  s1, s2, e = 0, 0,  math.exp(1)
  for _,col in pairs(y) do
     a  = norm(col, i.cells[col.at])
     b  = norm(col, j.cells[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

function dist(i,j)
  for _,col in pairs(i.context.cols.x) do
    a,b = i.cells[col.at], j.cells[col.at]
    inc = a=="?" and b=="?" and 1 or c.nump and gap(c,a,b) or (a==b and 0 or 1)
    d   = d + inc^the.p end
  return (d / (#i.context.cols.x)) ^ (1/the.p) end
-------------------------------------------------------------------------------
function COLS(names,      head,row,i)
  i={names=t, all={}, y={}, x={}}
  for at,txt in pairs(names) do
    col       = push(i.all, (txt:find"^[A-Z]" and NUM or SYM)(at, txt))
    col.goap = txt:find"[-+!]$"
    if not txt:find":$" then
      push(col.goalp, i.y or i.x, col) end end
  return i end
-------------------------------------------------------------------------------
function EGS(names) return {rows={}, cols=COLS(names)} end
function eg(i,row)
  cells = push(i.rows, row.cells and row or ROW(i,row)).cells
  for n,c in pairs(i.cols.all) do (c.nump and num or sym)(c, cells[n]) end end

function mid(i,cols)
  cols = cols or i.cols.y
  return map(cols,function(col) return col.nump and col.mu or mode(col) end) end

function clone(i,rows,  j)
 j=EGS(i.cols.names);for _,row in pairs({} or rows) do eg(j,row) end;return j end
-------------------------------------------------------------------------------
local go,no={},{}

function thes(f1,f2,k,x)
  for n,flag in ipairs(arg) do if flag==f1 or flag==f2 then
    x = x=="false" and"true" or x=="true" and"false" or arg[n+1] end end
  the[k] = string2thing(x) end

function main(     tmp,defaults,ok,msg,fails)
  fails=0    -- this code will return number of failures
  help:gsub( -- parse help text for flags and defaults, check CLI for updates
          "\n ([-][^%s]+)[%s]+([-][-])([^%s]+))[^\n]*%s([^%s]+)",thes)
  if the.help -- then pretty print help text
  then  print(help:gsub("%u%u+", "\27[31m%1\27[0m")
              :gsub("(%s)([-][-]?[^%s]+)(%s)","%1\27[33m%2\27[0m%3"),"")
  else
    tmp, defaults = {},{}
    for k,f in pairs(go) do if type(f)=="function" then push(tmp,k) end end
    for k,v in pairs(the) do defaults[k]=v end
    if go[the.todo] then tmp={the.todo} end
    for _,one in pairs(sort(tmp))  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
      status = go[one]()                              -- run demo
      if status ~= true then
        print("-- Error",one,status)
        fails = fails + 1 end end end               -- update fails
    for k,v in pairs(_ENV) do
      if not b4[k] then print("?",k,type(v)) end end -- list any rogue variables
    os.exit(fails) end                              -- exit status == fails

function go.num(     n,mu,sd)
  n, mu, sd = Num(), 10, 1
  for i=1,10^3 do
    n:update(mu + sd*math.sqrt(-2*math.log(r()))*math.cos(2*math.pi*r())) end
  ok(abs(n:mid() - mu) < 0.025, "sd")
  ok(abs(n:div() - sd) < 0.05,  "div")  end

main()
```

```lua
-------------------------------------------------------------------------------
function SOME() return {all={}, ok=false, n=0} end
function some(i,x)
  if x=="?" then return x end
  i.n = 1 + i.n
  if     #i.all < the.some      then i.ok=false; push(i.all, x)
  elseif rand() < the.some/i.n then i.ok=false; i.all[rand(#i.all)]=x end end

function per(i,p)
  i.all = i.ok and i.all or sort(i.all); i.ok=true
  return i.all[math.max(1, math.min(#i.all, (p or .5)*#i.all//1))] end
```