```lua
local the, help = {}, [[
ween.lua [OPTIONS]
ween (vb), archaic. To think or imagine.

A small sample multi-objective optimizer / data miner.
(c)2021 Tim Menzies <timm@ieee.org> unlicense.org

OPTIONS:
  -best   X  Best end  of the examples.              = .5
  -debug  X  Run one test, show stack dumps on fail.  = ing
  -file   X  Read data from files.                    = ../../data/auto93.csv
  -h         Show help.                               = false
  -hints  X  How many  to evaluate each iteration.    = 4
  -p      X  Coefficient on distance calculation.     = 2
  -seed   X  Random number seed.                      = 10019
  -todo   X  Demos to run at start-up. 'all'=run all. = ing]]

local b4={}; for k,_ in pairs(_ENV) do b4[k]=k end
local function rogues()
  for k,v in pairs(_ENV) do if not b4[k] then print("?:",k,type(v)) end end end
-------------------------------------------------------------------------------
local pop,csv,fmt,map,keys,sort,copy,norm,push
local color,first,firsts,coerce,second22,shuffle,bchop
fmt = string.format
function coerce(x)      return tonumber(x) or x end
function color(n,s)     return fmt("\27[1m\27[%sm%s\27[0m",n,s) end
function copy(t,  u)    u={};for k,v in pairs(t) do u[k]=v    end; return u end
function keys(t,u)      u={};for k,_ in pairs(t) do u[1+#u]=k end; return sort(u);end
function first(x)       return x[1] end
function firsts(x,y)    return x[1] < y[1] end
function norm(lo,hi,x) return math.abs(lo-hi)<1E-9 and 0 or (x-lo)/(hi-lo) end
function pop(t)         return table.remove(t) end
function push(t,x)      table.insert(t,x); return x end
function second22(_,t)      return t[2] end
function sort(t,f)      table.sort(t,f); return t end
function shuffle(t,    j)
  for i=#t,2,-1 do j=randi(1,i); t[i],t[j]=t[j],t[i] end; return t end
function map(t,f,   u)
  u,f = {},f or same; for k,v in pairs(t) do push(u, f(k,v)) end; return u end

function csv(file)
  file = io.input(file)
  return function(    t,x)
    x = io.read()
    if x then
      t={};for y in x:gsub("%s*",""):gmatch"([^,]+)" do push(t,coerce(y)) end
      if #t>0 then return t end
    else io.close(file) end end end

local shout,out
function shout(x) print(out(x)) end
function out(t,      u,key,val)
  function key(_,k) return string.format(":%s %s", k, out(t[k])) end
  function val(_,v) return out(v) end
  if type(t) ~= "table" then return tostring(t) end
  u = #t>0 and map(t, val) or map(keys(t), key)
  return "{"..table.concat(u," ")..'}' end

function bchop(t,val,policy,      lo,hi,mid)
  lt = lt or function(x,y) return x < y end
  lo,hi = 1,#t
  while lo <= hi do
    mid =(lo+hi) // 2
    if policy(t[mid],val) then lo=mid+1 else hi= mid-1 end end
  return math.min(lo,#t)  end

local randi,rand,Seed -- remember to set seed before using this
function randi(lo,hi) return math.floor(0.5 + rand(lo,hi)) end
function rand(lo,hi)
  lo, hi = lo or 0, hi or 1
  Seed = (16807 * Seed) % 2147483647
  return lo + (hi-lo) * Seed / 2147483647 end
-------------------------------------------------------------------------------
local slurp,sample,dist,ordered,hint,left_is_best
function slurp(  i) for eg in csv(the.file) do i=sample(i,eg) end; return i end

function sample(i,eg)
  local numeric,independent,dependent,head,data,datum
  i = i or {xs={},nys=0,ys={},lo={},hi={},w={},egs={},heads={},divs={}}
  function head(n,x)
    function numeric()     i.lo[n]= math.huge; i.hi[n]= -i.lo[n] end
    function independent() i.xs[n]= x end
    function dependent()
      i.w[n]  = x:find"-" and -1 or 1
      i.ys[n] = x
      i.nys   = i.nys+1 end
    if not x:find":" then
      if x:match"^[A-Z]" then numeric() end
      if x:find"-" or x:find"+" then dependent() else independent() end end
    return x end
  function datum(n,x)
    if x ~= "?" then
      if i.lo[n] then
        i.lo[n] = math.min(i.lo[n],x)
        i.hi[n] = math.max(i.hi[n],x) end end
    return x end
  if #i.heads==0 then i.heads=map(eg,head) else push(i.egs,map(eg,datum)) end
  return i end
-------------------------------------------------------------------------------
function left_is_best(i,left,right,     a,b,lefts,rights)
  lefts,rights=0,0
  for n,_ in pairs(i.ys) do
    a  = norm(i.lo[n], i.hi[n], left[n])
    b  = norm(i.lo[n], i.hi[n], right[n])
    lefts  = lefts  - 2.71828^(i.w[n] * (a-b)/i.nys)
    rights = rights - 2.71828^(i.w[n] * (b-a)/i.nys) end
  return lefts/i.nys < rights/i.nys end

function ordered(i,egs)
  return sort(egs or i.egs, function(a,b) return left_is_best(i,a,b) end) end

function dist(i,eg1,eg2)
  local function dist1(lo,hi,a,b)
  if   lo
  then if     a=="?" then b=norm(lo,hi,b); a = b>.5 and 0 or 1
       elseif b=="?" then a=norm(lo,hi,a); b = a>.5 and 0 or 1
       else               a,b = norm(lo,hi,a), norm(lo,hi,b) end
       return math.abs(a-b)
  else return a==b and 0 or 1 end
  end ------------------
  local d,n = 0,0
  local a,b,inc
  for col,_ in pairs(i.xs) do
    a,b = eg1[col], eg2[col]
    inc = a=="?" and b=="?" and 1 or dist1(i.lo[col],i.hi[col],a,b)
    d   = d + inc^the.p
    n   = n + 1 end
  return (d/n)^(1/the.p) end

function hint(i,egs)
  local function hint1(egs, all, min, evals,lvl)
    local scoreds,nearest,best = {}
    function nearest(_,eg, tmp)
      return sort(map(scoreds, function(rank,scored)
                 return {rank+dist(i,eg,scored)/10^6, eg} end),firsts)[1] end
    if   #egs <= 2*min
    then scoreds = egs
         return map(sort(map(all,nearest),firsts), second22)
    else for j=1,the.hints do push(scoreds, pop(egs)) end
         best, scoreds = {}, ordered(i,scoreds)
         egs = sort(map(egs,nearest),firsts)
         for j=1,(#egs)//2 do push(best, egs[j][2]) end
         return hint1(best, all, min, evals+the.hints,lvl..".") end
  end ------------------
  egs = egs or i.egs
  return hint1( copy(shuffle(egs)), egs, (#egs)^the.best, 0,"") end
```



```lua
-------------------------------------------------------------------------------
local go={}
function go.ing() return true end
function go.the() shout(the) end
function go.csv(  n)
  n=0; for eg in csv(the.file) do n=n+1; if n>390 then shout(eg) end end end
function go.lib(    u,t)
  t= {10,20,30,40}
  u= copy(shuffle(t))
  t[1]=100
  assert(u[1] ~= t[1])
  assert(u[1] ~= 100) end

function go.sample(  s)
  s=slurp()
  assert(398 == #s.egs)
  assert(3 == s.lo[1]) end

function go.ordered(  _,i,egs)
  i = slurp()
  egs = ordered(i)
  shout(i.heads)
  for j=1,5 do shout(egs[j]) end
  print("#")
  for j=#egs-5,#egs do shout(egs[j]) end end

function go.dist(  i,dist1,t)
  function dist1(_,eg) return {dist(i,i.egs[1],eg), eg} end
  i = slurp()
  t=map(i.egs,dist1)
  for j=1,5 do print(j,fmt("%5.3f",t[j][1]),out(t[j][2])) end
  print("#")
  for j=(#t)-5,#t do print(j,fmt("%5.3f",t[j][1]),out(t[j][2])) end end

function go.hint(  i,sort1,sort2,s,lt)
  function lt(a,b) return left_is_best(i,a,b) end
  i=slurp()
  sort1= ordered(i)
  sort2= hint(i)
  for m,eg in pairs(sort2) do
     if m < 20 then shout(eg) end
     if m > (#sort2)-20 then shout(eg) end end end
-------------------------------------------------------------------------------
-- Run demos, each time resetting random seed and the global config options.
-- Return to the operating system then number of failing demos.
local function main(the,go)
  local no,defaults = 0,{}
  for k,v in pairs(the) do defaults[k]=v end
  local function reset(x)
    for k,v in pairs(defaults) do the[k]=v end
    Seed = the.seed or 10019 end
  reset()
  go[ the.debug ]()
  for _,it in pairs(the.todo=="all" and keys(go) or {the.todo}) do
    if type(go[it]) ~= "function" then return print("NOFUN:",it) end
    reset()
    local ok,msg = pcall( go[it] )
    if ok then print(color(32,"PASS ")..it)
          else print(color(31,"FAIL ")..it,msg); no=no+1 end end
  rogues()
  os.exit(no) end
-------------------------------------------------------------------------------
-- Make 'the' options array from help string and any updates from command line.
(help or ""):gsub("^.*OPTIONS:",""):gsub("\n%s*-([^%s]+)[^\n]*%s([^%s]+)",
   function(flag,x)
     for n,word in ipairs(arg) do if word=="-"..flag then
       x = x=="false" and "true" or tonumber(arg[n+1]) or arg[n+1] end end
     if x=="false" then x=false elseif x=="true" then x=true end
     the[flag]=x end)

if the.h then print(help) else main(the,go) end
```