```lua
--- vim: ts=2 sw=2 et:
-- ## Coding concentions
--
-- - Code 80 chars wide, or less.  Functions in 1 line, if you can.
--   Indent with two spaces. Divide code into 120 line (or less) pages.
-- - Minimize use of local (exception: define all functions as local
--   at top of file).
-- - Use polymorphic but not inheritance (simpler debugging).
--   Use UPPERCASE for class names. All classes new a `new` constructor.
-- - Use `i` instead of `self`. Use `_` to denote the last created class/
--   Use `__` for anonymous variable.s
-- - Set flags in help string top of file. Allow for `-h` on the command line
--   to print help
-- - The `go` functions store tests. tests should be silent unless they
--   fail tests can be disabled by renaming from `go.fun` to `no.fun`.
--   Those tests should return `true` if the test passes.
--   On exit, return number of failed tests.
--
-- ## About the learning
--
-- - Beware missing values (marked in "?") and avoid them
-- - Where possible all learning should be  incremental.
-- - Isolate operating system interaction.
--------------------------------------------------------------------------------
local b4,help = {},[[
SAW2: 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 saw2.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
  -r  --rnd   rounding rules           = %5.2f
  -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. ]]
--------------------------------------------------------------------------------
-- ## Namespace
local the={}
local _,big,clone,csv,demos,discretize,dist,eg,entropy,fmt,gap,like,lt
local map,merged,mid,mode,mu,norm,num,o,obj,oo,pdf,per,push
local rand,range,rangeB4,rnd,rnds,rowB4,slice,sort,some,same,sd,string2thing,sym,t
hese
local NUM,SYM,RANGE,EGS,COLS,ROW
for k,__ in pairs(_ENV) do b4[k]=k end -- At end, use `b4` to find rogue vars.
--------------------------------------------------------------------------------
-- ## Utils
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 lt(x)        return function(a,b) return a[x] < b[x] end end
function slice(t,i,j,k,     u)
  i,j = i or 1,j or #t
  k   = (k or 1)
  k   = (j - i)/n
  u={}; for n=i,j,k do u[1+#u] = t[n] 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 rnds(t,f) return map(t, function(x) return rnd(x,f) end) end
function rnd(x,f)
  return fmt(type(x)=="number" and (x~=x//1 and f or the.rnd) or"%s",x) end

function obj(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
  _ = t
  return setmetatable(t, {__call=new}) end
--------------------------------------------------------------------------------
NUM=obj"NUM"
function _.new(i,at,txt)
  i.at=at or 0; i.txt=txt or ""; i.lo,i.hi=big, -big
  i.n,i.mu,i.m2,i.sd = 0,0,0,0,0;  i.w=(txt or""):find"-$" and -1 or 1 end

function _.add(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 _.bin(i,x,n,  b) b=(i.hi-i.lo)/n; return math.floor(x/b+0.5)*b end
function _.mid(i) return i.mu end

function _.norm(i,x) return i.hi-i.lo<1E-9 and 0 or (x-i.lo)/(i.hi-i.lo+1/big)end

function _.dist(i, x,y)
  if     x=="?" and y=="?" then return 1 end
  if     x=="?"            then y = i:norm(y); x = y<.5 and 1 or 0
  elseif y=="?"            then x = i:norm(x); y = x<.5 and 1 or 0
  else x,y = i:norm(x), i:norm(y) end
  return math.abs(x - y) end

function _.like(i,x,__,       e)
  return (x < i.mu - 4*i.sd and 0 or x > i.mu + 4*i.sd and 0 or
    2.7183^(-(x - i.mu)^2 / (z + 2*i.sd^2))/(z + (math.pi*2*i.sd^2)^.5)) end
--------------------------------------------------------------------------------
SYM=obj"SYM"
function _.new(i,at,txt) i.at=at or 0; i.txt=txt or ""; i.n,i.all = 0,{} end
function _.add(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 _.dist(i,x,y) return (a==b and 0 or 1) end

function _.mid(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 _.div(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 _.like(i,x,prior) return ((c.all[x] or 0) + the.m*prior)/(c.n+the.m) end
--------------------------------------------------------------------------------
RANGE=obj"RANGE"
function _.new(i,col,lo,hi,y)
  i.cols, i.x, i.y = col, ({lo=lo or big, hi=hi or -big}), (y or  SYM()) end

function _.add(i,x,y)
  if x=="?" then return x end
  i.x.lo = math.min(i.x.lo,x)
  i.x.hi = math.max(i.x.hi,x)
  i.y:add(x,y) end

function _.__lt(i,j) return i.col.at == j.col.at and i.x.lo < j.x.lo end
function _.of(i,x)   return i.y.all[x] or 0 end

function _.selects(i,t,     x)
  t = t.cells and t.cells or t
  x = t[i.at]
  return x=="?" or (i.x.lo==i.x.hi and i.x.lo==x) or (i.x.lo<=x and x<i.x.hi)end

function _.__tostring(i)
  local x, lo, hi = i.txt, i.x.lo, i.x.hi
  if     lo ==  hi  then return fmt("%s == %s",x, lo)
  elseif hi ==  big then return fmt("%s >= %s",x, lo)
  elseif lo == -big then return fmt("%s < %s", x, hi)
  else                   return fmt("%s <= %s < %s",lo,x,hi) end end

function _.merged(i,j,n0,    k)
  if i.at == j.at then
    k = SYM(i.y.at, i.y.txt)
    i,j = i.y, j.y
    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.y.n<(n0 or 0) or j.y.n<(n0 or 0) or (ent(i)*i.n+ent(j)*j.n)/k.n > ent(k)
    then return RANGE(i.col, i.lo, j.hi, k) end end end
--------------------------------------------------------------------------------
ROW=obj"ROW"
function _.new(i,eg, cells) i.base,i.cells = eg,cells end
function _.__lt(i,j,     s1,s2,e,y,a,b)
  y = i.base.cols.y
  s1, s2, e = 0, 0,  math.exp(1)
  for __,col in pairs(y) do
     a  = col:norm(i.cells[col.at])
     b  = col:norm(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 _.__sub(i,j)
  for __,col in pairs(i.base.cols.x) do
    a,b = i.cells[col.at], j.cells[col.at]
    inc = a=="?" and b=="?" and 1 or col:dist(a,b)
    d   = d + inc^the.p end
  return (d / (#i.base.cols.x)) ^ (1/the.p) end

function _.around(i,rows)
  return sort(map(rows or i.base.rows, function(j) return {dist=i-j,row=j} end),
              lt"dist") end
--------------------------------------------------------------------------------
COLS=obj"COLS"
function _.new(i,names,     head,row,col)
  i.names=names; i.all={}; i.y={}; i.x={}
  for at,txt in pairs(names) do
    col       = push(i.all, (txt:find"^[A-Z]" and NUM or SYM)(at, txt))
    col.goalp = txt:find"[!+-]$" and true or false
    if not txt:find":$" then
      if txt:find"!$" then i.klass=col end
      push(col.goalp and i.y or i.x, col) end end end
--------------------------------------------------------------------------------
EGS=obj"EGS"
function _.new(i,names) i.rows,i.cols = {}, COLS(names) end
function _.load(f,   i)
  for row in csv(the.file) do if i then i:add(row) else i=EGS(row) end end
  return i end

function _.add(i,row,    cells)
  cells = push(i.rows, row.cells and row or ROW(i,row)).cells
  for n,col in pairs(i.cols.all) do col:add(cells[n]) end end

function _.mid(i,cols)
  return map(cols or i.cols.y, function(c) return c:mid() end) end

function _.copy(i,rows,  j)
  j=EGS(i.cols.names); for __,r in pairs(rows or {}) do j:add(r) end;return j end

function _.like(i,t,overall, nHypotheses,      c)
  prior = (#i.rows + the.k) / (overall + the.k * nHypotheses)
  like  = math.log(prior)
  for at,x in pairs(t) do
    c=i.cols.all.at[at]
    if x~="?" and not c.goalp then
      like = math.log(col:like(x)) + like end end
  return like end
```

```lua

local _merge, _xpand, _ranges
function _.ranges(i,one,two,   t)
  t={}; for _,c in pairs(i.cols.x) do t[c.at]=_ranges(c,one,two) end;return t end

function _ranges(col,yes,no,    out,x,d)
  out = {}
  for _,what in pairs{{rows=yes, klass=true}, {rows=no, klass=false}} do
    for _,row in pairs(what.rows) do x = row.cells[col.at]; if x~="?" then
      d = col:discretize(x,the.bins)
      out[d] = out[d] or RANGE{col,x,x}
      out[d]:add(x, what.klass) end end end
  return _xpand(_merge(sort(out)))  end

function _merge(b4,        a,b,c,j,n,tmp)
  j,n,tmp = 1,#b4,{}
  while j<=n do
    a, b = b4[j], b4[j+1]
    if b then c = a:merged(b); if c then a,j = c,j+1 end end
    tmp[#tmp+1] = a
    j = j+1 end
  return #tmp==#b4 and tmp or _merge(tmp) end

function _xpand(t)
  for j=2,#t do t[j].lo=t[j-1].hi end; t[1].lo, t[#t].hi= -big,big; return t end
```

```lua

-------------------------------------------------------------------------------
local go,no={},{}

function these(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 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 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
  return fails end                               -- return total failure count

function go.the()     return type(the.bins)=="number" end
function go.sort(  t) return 0==sort({100,3,4,2,10,0})[1] end

function go.num(     n,mu,sd)
  n, mu, sd = NUM(), 10, 1
  for i=1,10^4 do
    n:add(mu+sd*math.sqrt(-2*math.log(rand()))*math.cos(2*math.pi*rand())) end
  return math.abs(n.mu - mu) < 0.05 and math.abs(n.sd - sd) < 0.5 end

function go.rows( n,m)
  m,n=0,0; for row in csv(the.file) do m=m+1; n=n+#row; end; return n/m==8 end

function go.cols(  i)
  i=COLS{"name","Age","ShoeSize-"}
  return i.y[1].w == -1 end

function go.egs(  it)
  it = EGS.load(the.file); return math.abs(2970 - it.cols.y[1].mu) < 1 end

function go.ranges(  it,n,a,b)
  it = EGS.load(the.file)
  print(oo(rnds(it:mid())))
  it.rows = sort(it.rows)
  n = (#it.rows)^.5
  a,b = slice(it.rows,1,n), slice(it.rows,n+1,#it.rows,3*n)
  print(o(rnds(it:copy(a):mid())), o(rnds(it:copy(b):mid())))
  --oo(a:mid())
  --oo(b:mid())
  return math.abs(2970 - it.cols.y[1].mu) < 1 end
--------------------------------------------------------------------------------
help:gsub(  -- parse help text for flags and defaults, check CLI for updates
         "\n ([-][^%s]+)[%s]+([-][-]([^%s]+))[^\n]*%s([^%s]+)",these)
if the.help then
  print(help:gsub("%u%u+", "\27[31m%1\27[0m")
            :gsub("(%s)([-][-]?[^%s]+)(%s)","%1\27[33m%2\27[0m%3"),"")
else
  local status = demos()
  for k,v in pairs(_ENV) do if not b4[k] then print("?",k,type(v)) end  end
  os.exit(status) end
--------------------------------------------------------------------------------
-- 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
```