```lua
#!/usr/bin/env lua
-- vim: filetype=lua ts=2 sw=2 et:
-- (c) 2022, Tim Menzies,  opensource.org/licenses/Fair
-- Les Å“uvres peuvent Ãªtre rÃ©utilisÃ©es Ã  condition d'Ãªtre accompagnÃ©es du
-- texte de cette licence, afin que tout utilisateur en soit informÃ©.
-- AVERTISSEMENT : LES Å“UVRES N'ONT AUCUNE GARANTIE.
local b4={}; for k,v in pairs(_ENV) do b4[k]=v end
local any, coerce, csv, ent, fails, fmt, fu, go, id, lt, many, map, obj, push
local no, o, oo, ok, per, r, rnd, rnds, same, sd, sort, sum, the, work1, work
local the, help={}, [[
wicked: explore the world better,  explore the world for good.
(c) 2022, Tim Menzies, opensource.org/licenses/Fair

  .-------.
  |       |
  |  Ba   |   Bad <----.  planning = (better - bad)
  |   56  |            |  monitor  = (bad - better)
  .-------.------.     |
          |      |     v
          |  Be  |
          |   4  | Better
          .------.

USAGE:
  wicket.lua [OPTIONS]

OPTIONS:
  --K       -K  manage low class counts     = 1
  --M       -M  manage low evidence counts  = 2
  --far     -F  how far to go for far       = .9
  --p       -p  coefficient on distance     = 2
  --seed    -S  seed                        = 10019
  --some    -s  sample size for distances   = 512
  --stop    -T  how far to go for far       = 20
  --min     -m  size of min space           = .5

OPTIONS (other):
  --dump    -d  dump stack+exit on error    = false
  --file    -f  file name                   = ../etc/data/auto93.csv
  --help    -h  show help                   = false
  --rnd     -r  rounding numbers            = %5.3f
  --todo    -t  start up action             = nothing ]]
-----------------------------------------------------------------
r   = math.random
fmt = string.format
function fu(x) return function(t) return t[x] end end

function lt(x) return function(t,u) return t[x] < u[x] end end
function sort(t,f) table.sort(t,type(f)=="string" and lt(f) or f);return t end

function push(t,x)    t[1+#t]=x; return x end
function map(t,f, u) u={};for _,v in pairs(t) do u[1+#u]=f(v) end;return u end
function sum(t,f, u) u=0; for _,v in pairs(t) do u=u+f(v)      end;return u end

function any(a, i)     i=r()*#a//1; i=math.max(1,math.min(i,#a)); return a[i] end
function many(a,n, u) u={};for j=1,n do push(u,any(a)) end;return u end

function same(x) return x end
function sd(t,f)  f=f or same; return (f(per(t,.9)) - f(per(t,.1)))/2.56 end
function per(t,p) return t[ ((p or .5)*#t) // 1 ] 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 oo(t)  print(o(t)) end
function o(t,      u,one)
  one= function(k,v) return #t>0 and tostring(v) or fmt(":%s %s",k,v) end
  u={}; for k,v in pairs(t) do u[1+#u] = one(k,v) end
  if #t==0 then sort(u) end
  return (t.is or "").."{"..table.concat(u,"")..."}" end

function coerce(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 row[1+#row]=coerce(x) end
      return row end end end

function work1(x,    b4)
  b4={}; for k,v in pairs(the) do b4[k]=v end
  math.randomseed(the.seed)
  if go[x] then print(x); go[x]() end
  for k,v in pairs(b4) do the[k]=v end end

function work(   t)
  t={}; for k,_ in pairs(go) do push(t,k) end
  for _,x in pairs(sort(t)) do work1(x)  end  end
-----------------------------------------------------------------
local _id=0
function id() _id = _id+1; return _id end

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

```lua
-----------------------------------------------------------------
local Bin=obj"Bin"
function Bin:new(txt,at,n, lo,hi,ystats)
  self.at, self.txt, self.n = at, txt, n
  self.lo, self.hi, self.ystats = lo, hi, ystats end

function Bin:__tostring()
  local x,lo,hi,big = self.name, self.lo, self.hi, math.huge
  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 Bin:select(row)
  local x, lo, hi = row[self.at], self.lo, self.hi
  return x=="?" or lo == hi and lo == x or lo <= x and x < hi end
-----------------------------------------------------------------
local Sym=obj"Sym"
function Sym:new(at,txt)
  self.at  = at or 0
  self.txt = txt or ""
  self.n   = 0
  self.has, self.mode, self.most = {},nil,0 end

function Sym:sub(x) return self:add(x,-1) end

function Sym:add(x,inc)
  if x ~= "?" then
    inc = inc or 1
    self.n = self.n + inc
    self.has[x] = (self.has[x] or 0) + inc
    if self.has[x] > self.most then self.most,self.mode = self.has[x],x end end
  return x end

function Sym:mid() return self.mode end
function Sym:div(  e)
  e=0;for _,m in pairs(t) do e=e-m/self.n*math.log(m/self.n,2); return e end end

function Sym:dist(x,y) return x=="?" and y=="?" and 1 or x==y and 0 or 1 end

function Sym:bins(left,right,     out,has,n)
  n,out = 0,{}
  function has(x)
    n=n+1
    out[x] = out[x] or Bin(self.at, self.txt, n, x, x, Sym()) end
  for _,r in pairs(left)  do has(x); out[x].ystats:add(1) end
  for _,r in pairs(right) do has(x); out[x].ystats:add(0) end
  return map(out, function(x) return x end) end
-----------------------------------------------------------------
local Num=obj"Num"
function Num:new(at,txt)
  self.at  = at or 0
  self.txt = txt or ""
  self.n, self.mu, self.m2 = 0,0,0
  self.w   = self.txt:find"-$" and -1 or 1
  self.lo, self.hi = math.huge, -math.huge end

function Num:add(x,        d)
  if x ~="?" then
    self.n  = self.n + 1
    self.lo = math.min(x, self.lo)
    self.hi = math.max(x, self.hi)
    d       = x - self.mu
    self.mu = self.mu + d/self.n
    self.m2 = self.m2 + d*(x - self.mu) end
  return x end

function Num:mid() return self.mu end
function Num:div() return (self.m2/(self.n - 1))^0.5 end

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

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

function Num:bins(left,right,        xy,out,recurse,div,xy,epsilon,small)
  function div(lo,hi,       cut,lhs,rhs,best,b4,x0,x,xmax,y,t,stats)
    lhs, rhs, ystats = Sym(), Sym(), Sym()
    for i=lo,hi do ystats:add( rhs:add(xy[i].y) ) end
    best = rhs:div()
    for i=lo,hi do
      x, y = xy[i].x, xy[i].y
      lhs:add(y)
      rhs:sub(y)
      if lhs.n>small and rhs.n>small then
        if x - xy[lo].x > epsilon and xy[hi].x - x > epsilon then
          if x ~= xy[i+1].x then
            tmp = (lhs.n*lhs:div() + rhs.n*rhs:div())  / (lhs.n + rhs.n)
            if tmp*.95 < best then
              best,cut = tmp,i end end end end end
    return cut, ystats
  end ----------------
  function recurse(lo,hi,    cut,systats)
    cut, ystats = div(lo,hi)
    if   cut
    then recurse(lo,    cut)
         recurse(cut+1, hi)
    else b4=push(out,
                 Bin(self.txt, self.at, 1+#out, b4, xy[hi].x, ystats)).hi end
  end ---------------------------
  b4, xy, out = -math.huge, {}, {}
  for _,r in pairs(left)  do if x ~="?" then push(xy,{x=r.cells[c],y=1}) end end
  for _,r in pairs(right) do if x ~="?" then push(xy,{x=r.cells[c],y=0}) end end
  xy       = sort(xy, lt"x")
  epsilon = sd(xy, fu"x")*the.cohen
  small   = (#xy)^the.min
  recurse(1,#xy)
  out[#out].hi = math.huge
  return out end
```

```
223 ------------------------------------------------------------------------
224 local Cols=obj"Cols"
225 function Cols:new(names,    col)
226   self.names, self.all, self.x, self.y, self.klass = names, {}, {}, {}, nil
227   for at,txt in pairs(names) do
228     col = push(self.all, (txt:find"^[A-Z]" and Num or Sym)(at,txt))
229     if not txt:find"!$"  then
230       if txt:find"!$" then self.klass=col end
231       col.indep = not txt:find"[-+!]$"
232       push(col.indep and self.x or self.y, col) end end  end
233
234 function Cols:add(row)
235   for _,col in pairs(self.all) do col:add(row[col.at]) end
236   return row end
237
238 ------------------------------------------------------------------------
239 local Row=obj"Row"
240 function Row:new(t) self.cells = t end
241
242 ------------------------------------------------------------------------
243 local Egs=obj"Egs"
244 function Egs:new() self.rows,self.cols = {}, nil end
245
246 function Egs:clone(rows,    out)
247   out = Egs():add(self.cols.names)
248   for _,row in pairs(rows or {}) do out:add(row) end
249   return out end
250
251 function Egs:load(file)
252   for row in csv(file) do self:add(row) end; return self end
253
254 function Egs:add(t)
255   t = t.cells and t.cells or t
256   if   self.cols
257   then push(self.rows, Row(self.cols:add(t)))
258   else self.cols=Cols(t) end
259   return self end
260
261 function Egs:better(row1,row2)
262   local s1, s2, n, e = 0, 0, #self.cols.y, math.exp(1)
263   for _,col in pairs(self.cols.y) do
264     local a = col:norm(row1.cells[col.at])
265     local b = col:norm(row2.cells[col.at])
266     s1    = s1 - e^(col.w * (a - b) / n)
267     s2    = s2 - e^(col.w * (b - a) / n) end
268   return s1 / n < s2 / n  end
269
270 function Egs:betters(rows)
271   return sort(rows or self.rows, function(a,b) return self:better(a,b) end) end
272
273 function Egs:mid(cols)
274   return rnds(map(cols or self.cols.y, function(col) return col:mid() end)) end
275
276 function Egs:dist(row1,row2,    d,n)
277   d = sum(self.cols.x, function(col)
278                 return col:dist(row1.cells[col.at], row2.cells[col.at])^the.p end)
279   return (d / (#self.cols.x)) ^ (1/the.p) end
280
281 function Egs:around(row1, rows,    around)
282   function around(row2) return {dist=self:dist(row1,row2),row=row2} end
283   return sort(map(rows or self.rows,around), lt"dist") end
284
285 function Egs:far(row, rows)
286   return per(self:around(row, rows or many(self.rows,the.some)),the.far).row end
287
288 function Egs:unsuper(n,     recurse,known,rows,used,rest)
289   function known(row) used[row.id]=true; return row end
290   function recurse(rows,some,x,       y,best,a,b,c)
291     if #rows <= 20 then
292       oo(self:clone(rows):mid())
293     else
294       x = known( x or self:far(any(some),some))
295       y = known(     self:far(x,some))
296       if self:better(y, x) then io.write("/"); x,y = y,x else io.write(".") end
297       c = self:dist(x,y)
298       best = {}
299       for _,r in pairs(rows) do
300         a,b= self:dist(r,x), self:dist(r,y); r.x = (a^2+ c^2-b^2) / (2*c) end
301       for i,row in pairs(sort(rows, lt"x")) do
302         push(i < #rows//2 and best or rest,row) end
303       recurse(best, many(best,n), x)   end
304   end ---------------
305   used, rest = {}, {}
306   recurse(self.rows, many(self.rows,n)) end
307
308
```

```
308 ------------------------------------------------------------------------
309 fails,go,no = 0,{},{}
310 function ok(test,msg)
311   print("", test and "PASS"or "FAIL ", msg or "")
312   if not test then
313     fails= fails+1
314     if  the.dump then assert(test,msg) end end end
315
316 function go.many()
317   oo(many({10,20,30,40,50,60,70,80,90,100},3)) end
318
319 function go.unsuper(   eg,best)
320   eg = Egs():load(the.file)
321   oo(map(eg.cols.y, function(col) return col.txt end))
322   oo(map(eg.cols.y, function(col) return col.w end))
323   oo(eg:mid())
324   print("---")
325   for i=1,20 do eg:unsuper(128) end
326   eg:betters()
327   best = eg:clone()
328   for i=1,20 do best:add(eg.rows[i]) end
329   print("---")
330   oo(best:mid()) end
331
332 function go.eg1(   eg)
333   eg = Egs():load(the.file)
334   print(#eg.rows, eg.cols.y[1]) end
335
336 function go.dist(  eg,row2,t)
337   eg = Egs():load(the.file)
338   t={}; for i=1,20 do
339     row2= any(eg.rows)
340     push(t, {dist=eg:dist(eg.rows[1],row2), row = row2}) end
341   oo(eg.rows[1])
342   for _,two in pairs(sort(t,lt"dist")) do oo(two.row.cells) end end
343
344 function go.mids( eg,hi,lo,out)
345   eg = Egs():load(the.file)
346   oo(map(eg.cols.y, function(col) return col.txt end))
347   oo(map(eg.cols.y, function(col) return col.w end))
348   print("all",o(eg:mid()))
349   lo,hi = eg:clone(), eg:clone()
350   for i,row in pairs(eg:betters()) do
351     if i < 20             then lo:add(row) end
352     if i > #eg.rows - 20 then hi:add(row) end end
353   print("lo",o(lo:mid()))
354   print("hi",o(hi:mid())) end
355
356 ------------------------------------------------------------------------
357 help:gsub("\n ([-][-]([^%s]+))[%s]+(-[^%s]+)[^\n]*%s([^%s]+)",
358   function(long,key,short,x)
359     for n,flag in ipairs(arg) do
360       if flag==short or flag==long then
361         x = x=="false" and "true" or x=="true" and "false" or arg[n+1] end end
362     the[key] = coerce(x) end)
363
364 if the.help then print(help) end
365 if the.todo=="all" then work() else work1(the.todo) end
366 for k,v in pairs(_ENV) do if not b4[k] then print("?",k,type(v)) end end
367 os.exit(fails)
```