```lua
#!/usr/bin/env lua
-- vim: filetype=lua ts=2 sw=2 et:
---                            __               __
---                 __        /\ \             /\ \__
---      __  __  __ /\_\    ___\ \ \/'\      __ \ \ ,_\
---     /\ \/\ \/\ \\/\ \  /'___\ \ , <    /'__`\\ \ \/
---     \ \ \_/ \_/ \\ \ \/\ \__/\ \ \\`\ /\  __/ \ \ \_
---      \ \___x___/' \ \_\ \____\\ \_\ \_\ \____\ \ \__\
---       \/__//__/    \/_/\/____/ \/_/\/_/\/____/  \/__/

-- (c) 2022, Tim Menzies,  opensource.org/licenses/Fair
-- 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.

-- xxxx  kill cloning
-- add back here the shorter doc string and maom amd go.rogue

local b4={}; for k,v in pairs(_ENV) do b4[k]=v end
local any,coerce,csv,ent,fails,fmt,fu,go,id,lt,main,many,map,obj,push
local no,o,oo,ok,per,r,rnd,rnds,runDemo,same,sd,settings,shuffle,sort,sum
--------------------------------------------------------------------------------
local the, help={}, [[
wicket: explore the world better,  explore the world for good.
(c) 2022, Tim Menzies, opensource.org/licenses/Fair

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

USAGE:
  wicket.lua [OPTIONS]

OPTIONS:
  --cohen    -c  cohen                      = .35
  --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
  --best     -B  best percent               = .05

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 same(x) return x end
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 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 sort(t,f)   table.sort(t,f); return t end
function sum(t,f,n)
  n=0; for _,x in pairs(t) do n=n+(f or same)(x) end; return n end

function shuffle(t,   j)
  for i=#t,2,-1 do j=math.random(i); t[i],t[j]=t[j],t[i] end; return t 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)
  if n>=#a then return shuffle(a) end
  u={};for j=1,n do push(u,any(a)) end;return u 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

---     _|_ |_  o ._   _   _    _|_  _     _ _|_ ._ o ._   _   _
---      |_ | | | | | (_| _>     |_ (_)   _>  |_ |  | | | (_| _>
---                    _|                                  _|
function oo(t) print(o(t)) end
function o(t,    u,one,sorted)
  sorted = #t>0 -- true when array's indexes are 1,2...#t
  one= function(k,v) return sorted 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
  return (t.is or "").."{"..table.concat(sorted and u or sort(u)," ").."}" 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

---      _ _|_ ._ o ._   _   _    _|_  _    _|_ |_  o ._   _   _
---     _>  |_ |  | | | (_| _>     |_ (_)    |_ | | | | | (_| _>
---                      _|                                _|
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
---      ._ _   _. o ._
---      | | | (_| | | |

function main(todo,   all)
  all={}; for k,_ in pairs(go) do push(all,k) end
  all = the.todo=="all" and sort(all) or {todo}
  for _,x in pairs(all)  do runDemo(x) end
  for k,v in pairs(_ENV) do if not b4[k] then print("?",k,type(v)) end end
  os.exit(fails) end

function runDemo(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 settings(txt, d)
  d={}
  txt:gsub("\n ([-][-]([^%s]+))[%s]+(-[^%s]+)[^\n]*%s([^%s]+)",
  function(long,key,short,x)
    for n,flag in ipairs(arg) do
      if flag==short or flag==long then
        x = x=="false" and "true" or x=="true" and "false" or arg[n+1] end end
    d[key] = coerce(x) end)
  if d.help then print(txt) end
  return d end

--------------------------------------------------------------------------------
---     ____ ___  _ ____ ____ ___ ____
---     |  | |__]  | |___ |     |  [__
---     |__| |__] _| |___ |___  |  ___]

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

---      |_  o ._   _
---      |_) | | | _>

local Bin=obj"Bin"
function Bin:new(t)
  self.pos, self.txt, self.n, self.has = t.pos, t.txt, t.n, {}
  self.lo, self.hi, self.ystats = t.lo, t.hi, t.stats end

function Bin:__tostring()
  local x,lo,hi,big = self.txt, 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(t)
  t = t.cells and t.cells or t
  local x, lo, hi = t[self.pos], 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(pos,txt)
   self.pos  = pos 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(self.has) do
         if m>0 then e = e-m/self.n * math.log(m/self.n,2) end end
  return e end

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

function Sym:bins(rows,     x,n,out,has,tmp,inc)
  n,out,tmp = 0,{},{}
  function inc(x) n=n+1; return n end
  function has(x) tmp[x]=tmp[x] or Bin({txt=self.txt,  pos=self.pos, n=inc(x),
                                        lo=x ,hi=x, stats=Sym()}) end
  for _,r in pairs(rows) do
    x = r.cells[self.pos]; has(x); tmp[x].ystats:add(r.klass) end
  for _,x in pairs(tmp) do push(out, x) end
  return out end

---     ._       ._ _
---     | | |_| | | |

local Num=obj"Num"
function Num:new(pos,txt)
   self.pos  = pos 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

local _bins
function Num:bins(rows,      xy,f)
  function f(row, x)
    x=row.cells[self.pos]; if x~="?" then return {x=x,y=row.klass} end end
  xy = sort(map(rows,f),lt"x")
  return _bins(self.txt,self.pos,xy,sd(xy, fu"x")*the.cohen,(#xy)^the.min) end

function _bins(txt,pos,xy,epsilon,small,      div,b4,out)
  function div(lo,hi,        x,y,cut,lhs,rhs,tmp,best,overall)
    lhs, rhs, overall = Sym(), Sym(), Sym()
    for i=lo,hi do overall: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( rhs:sub( y) )
      if lhs.n > small and rhs.n > small then
        if x ~= xy[i+1].x then
          if x - xy[lo].x > epsilon and xy[hi].x - x > epsilon then
            tmp = (lhs.n*lhs:div() + rhs.n*rhs:div())  / (lhs.n + rhs.n)
            if tmp < best then
              best,cut = tmp,i end end end end end
    if   cut
    then div(lo,    cut)
         div(cut+1, hi)
    else b4 = push(out, Bin({txt=txt, pos=pos, n=1+#out,lo=b4,
                                hi=xy[hi].x, stats=overall})).hi end
  end -------------------
  b4, out = -math.huge, {}
  div(1,#xy)
  out[#out].hi = math.huge
  return out end
```

```
---      ._   _
---      |   (_)  \/\/

local Row=obj"Row"
function Row:new(t)
  self.evalauted, self.klass, self.cells = false,false,t end

---      _   _   |   _
---     (_  (_)  |  _>

local Cols=obj"Cols"
function Cols:new(names,    col)
  self.names, self.all, self.x, self.y, self.klass = names, {}, {}, {}, nil
  for pos,txt in pairs(names) do
    col = push(self.all, (txt:find"^[A-Z]" and Num or Sym)(pos,txt))
    if not txt:find":$"  then
      if txt:find"!$" then self.klass=col end
      col.indep = not txt:find"[-+!]$"
      push(col.indep and self.x or self.y, col) end end  end

function Cols:add(row)
  for _,col in pairs(self.all) do col:add(row[col.pos]) end end

---      _    _    _
---     (/_  (_|  _>
---           _|

local Egs=obj"Egs"
function Egs:new() self.rows,self.cols = {}, nil end

function Egs:clone(rows,   out)
  out = Egs():add(self.cols.names)
  for _,row in pairs(rows or {}) do out:add(row) end
  return out end

function Egs:load(file)
  for row in csv(file) do self:add(row) end; return self end

function Egs:add(t,  row)
  if   self.cols
  then row = t.cells and t or Row(t)
       self.cols:add(row.cells)
       push(self.rows, row)
  else self.cols=Cols(t) end
  return self end

function Egs:better(row1,row2)
  local s1, s2, n, e = 0, 0, #self.cols.y, math.exp(1)
  for _,col in pairs(self.cols.y) do
    local a = col:norm(row1.cells[col.pos])
    local b = col:norm(row2.cells[col.pos])
    s1      = s1 - e^(col.w * (a - b) / n)
    s2      = s2 - e^(col.w * (b - a) / n) end
  return s1 / n < s2 / n  end

function Egs:betters(rows)
  return sort(rows or self.rows, function(a,b) return self:better(a,b) end) end

function Egs:mid(cols)
  return rnds(map(cols or self.cols.y, function(col) return col:mid() end)) end

function Egs:dist(row1,row2,   d,n)
  d = sum(self.cols.x, function(col)
           return col:dist(row1.cells[col.pos], row2.cells[col.pos])^the.p end)
  return (d / (#self.cols.x)) ^ (1/the.p) end

function Egs:around(row1, rows,    around)
  function around(row2) return  {dist=self:dist(row1,row2),row=row2} end
  return sort(map(rows or self.rows,around), lt"dist") end

function Egs:far(row, rows)
  return per(self:around(row, rows or many(self.rows,the.some)),the.far).row end

function Egs:sway(rows,stop, x,rest,   some,y,best,a,b,c)
  rows = rows or self.rows
  rest = rest or {}
  stop = stop or 2*the.best*#self.rows
  if #rows <= stop then return rows,rest end
  some = many(rows,the.some)
  x    = x or self:far(any(some), some)
  y    =      self:far(x,         some)
  if self:better(y, x) then x,y = y,x end
  x.evaluated = true
  y.evaluated = true
  c = self:dist(x,y)
  for _,row in pairs(rows) do
    a,b = self:dist(row,x), self:dist(row,y)
    row.x = (a^2+c^2-b^2)/(2*c) end
  best = {}
  for i,row in pairs(sort(rows, lt"x")) do
    push(i<#rows//2 and best or rest, row) end
  return self:sway(best, stop, x,rest)  end

function Egs:leaves(rows,stop,leaves,   best,w,bw)
  leaves= leaves or {}
  rows  = rows or self.rows
  stop  = stop or 2*(#self.rows)^the.min
  print(1)
  function w(bin)   return bin.ystats.n/#rows * bin.ystats:div() end
  function bw(bins) return {bins=bins, worth=sum(bins,w)} end
  print(3)
  if #rows < stop then
    return push(leaves,self:clone(rows)) end
  print(3.1)
  tmp=map(self.cols.x,function(c) return bw(c:bins(rows))end)
  oo(tmp[1].bins[1].ystats.has)
  os.exit()
  best=sort(map(self.cols.x,function(c) return bw(c:bins(rows))end),lt"worth")[1]
  print(4)
  for _,row in pairs(rows) do
    for _,bin in pairs(best.bins) do
      if bin:select(row) then push(bin.has, row); break; end end end
  for _,bin in pairs(best.bins) do
      if #bin.has < #rows then bin.has= self:leaves(bin.has,stop,leaves)  end end
  return leaves end


```

```
------------------------------------------------------------------------------
---    ___  ____ _  _ ____ ____
---   |   \ |___ |\/| |  | [__
---   |__/  |___ |  | |__| ___]

fails,go,no = 0,{},{}
function ok(test,msg)
  print("", test and "PASS"or "FAIL", msg or "")
  if not test then
    fails = fails+1
    if the.dump then assert(test,msg) end end end

function go.sum(    t)
  print(sum({1,2,3},same)) end

function go.list(    t)
  t={}; for txt,_ in pairs(go) do if txt~="list" then push(t,txt) end end
  for _,txt in pairs(sort(t)) do print(fmt("lua wicket.lua -t %s",txt)) end end

function go.div(  s)
  s=Sym()
  for _,x in pairs{"a","a","a","a","b","b","c"} do s:add(x) end
  ok(math.abs(1.376 - s:div()) < 0.01, "ent") end

function go.symbins(  eg,rows)
  eg   = Egs():load(the.file)
  rows = eg:betters()
  for _,row in pairs(rows) do row.klass=false end
  for i=1,(#rows)^the.min do rows[i].klass=true end
  for _,col in pairs(eg.cols.x) do
    for k,v in pairs(col:bins(rows)) do print(v) end end end

function go.leaves(  eg,rows,s,tree)
  eg   = Egs():load(the.file)
  rows = eg:betters()
  for i=1,(#rows)*.2 do rows[i].klass=true end
  s=Sym()
  for _,row in pairs(rows) do s:add(row.klass) end
  for _,eg1 in pairs(eg:leaves(eg.rows,10)) do
     oo(eg1:mid()) end
  end
--
function go.many()
  oo(many({10,20,30,40,50,60,70,80,90,100},100)) end

function go.sway(   eg,best,guesses,rest)
  local  used = function(row) if row.evaluated then return true end end
  eg = Egs():load(the.file)
  print(eg:leaves())
  oo(map(eg.cols.y, function(col) return col.txt end))
  oo(map(eg.cols.y, function(col) return col.w   end))
  print("before",o(eg:mid()))
  best,rest = eg:sway()
  print("sway", o(eg:clone(best):mid()))
  print("evals",#map(eg.rows, used), #best, #rest)
  take2 ={}
  for _,row in pairs(best)               do row.klass=true;  push(take2,row) end
  for _,row in pairs(many(rest,3*#best)) do row.klass=false; push(take2,row) end
  oo(take2)
  eg:leaves(take2,5)
  -- for _,row in pairs(rest) do row.klass=false end
  -- for _,row in pairs(best) do row.klass=true end
  -- for _,row in pairs(many(rest,3*#best)) do push(best,row) end
  -- for _,eg1 in pairs(eg:leaves(best)) do
  --   print(
  -- for _,row in pairs(many(rest, 3*#guesses)) do push(
  -- best= eg:clone()
  -- for i,row in pairs(eg:betters()) do if i< the.best*#eg.rows then best:add(row) else break end end
  -- print("best",o(best:mid()))
  end

function go.eg1(   eg)
  eg = Egs():load(the.file)
  print(#eg.rows, eg.cols.y[1]) end

function go.far(  eg)
  eg = Egs():load(the.file)
  print(eg:far(eg.rows[1],eg.rows)) end

function go.around(  eg)
  eg = Egs():load(the.file)
  print(eg:around(eg.rows[1])) end

function go.dist(  eg,row2,t)
  eg = Egs():load(the.file)
  t={}; for i=1,20 do
    row2= any(eg.rows)
    push(t, {dist=eg:dist(eg.rows[1],row2), row = row2}) end
  oo(eg.rows[1].cells)
  print("---")
  for _,two in pairs(sort(t,lt"dist")) do oo(two.row.cells) end end

function go.mids( eg,hi,lo,out)
  eg = Egs():load(the.file)
  oo(map(eg.cols.y, function(col) return col.txt end))
  oo(map(eg.cols.y, function(col) return col.w end))
  print("all",o(eg:mid()))
  lo,hi = eg:clone(), eg:clone()
  for i,row in pairs(eg:betters()) do
    if i < 20            then lo:add(row) end
    if i > #eg.rows - 20 then hi:add(row) end end
  print("lo",o(lo:mid()))
  print("hi",o(hi:mid())) end

------------------------------------------------------------------------------
---    ____ ___ ____ ____ ___
---    [__   |  |__| |__/  |
---    ___]  |  |  | |  \  |

the = settings(help)
main(the.todo)

--            _   _   _
--           (_) (_) (_)
--           | | | | | |
--           |p| | | | |
--           |k| | | | |
--           |w| | | | |
--           | | | | | |
```

```
--           | | |_| | |
--          \ | |__\| |
--           \|_______|

```