```lua
-- If you understand "it", can you write "it" shorter?  Lets try.
-- E.G. how short can we write a multi-objective semi-supervised learner?<p>
-- &copy;2022 Tim Menzies.  [Github](http://github.com/timm/l5) <hr>
--
-- One of my most productive days was throwing away 1,000 lines of code.
-- -- Ken Thompson<p>
-- It is vain to do with more what can be done with less.
-- -- William Of Occam<p>
-- Every block of stone has a statue inside it.
-- And it is the task of the sculptor to discover it.
-- -- Michelangelo.<p>
-- The more you have, the more you are occupied.
-- The less you have, the more free you are.  <br>-- Mother Teresa<p>
-- <img width=200 align=left src=cup.png>
-- Simplicity is the ultimate sophistication. <br>-- Leonardo da Vinci<p>
-- Simplicity is prerequisite for reliability.<br>-- Edsger W. Dijkstra<p>
-- Less, but better.                               <br>-- Dieter Rams<p>
-- less, plz                                   <br>-- timm <p>
-- My heroes: [Jack Diederich](https://www.youtube.com/watch?v=o9pEzgHorH0)
-- | [Hilary Mason](https://boingboing.net/2017/06/30/next-level-regexp.html)<p>
local help= [[
shorter.lua : a multi-objective semi-supervised learner.
(c)2022 Tim Menzies, timm@ieee.org

OPTIONS:
  -b  --Bins   max number of bins        = 16
  -F  --Few    only keep a "Few" numbers = 256
  -k  --k      handle rare classes       =  1
  -m  --m      handle rare attributes    =  2
  -p  --p      distance coefficient      =  2
  -S  --small  small leaf size           = .5
  -w  --wait   wait before classifying   =

OPTIONS (other):
  -f  --file   file           = ../../data/auto93.csv
  -g  --go     start-up goal  = nothing
  -h  --help   show help      = false
  -s  --seed   seed           = 10019]]
--------------------------------------------------------------------------------
-- ## Names
local _ = require"lib"
local argmax,big                   = _.argmax, _.big
local cli,csv,demos,klass,normpdf = _.cli,     _.csv,  _.demos,_.klass, _.normpdf
local oo,push,read,rnd,same,str   = _.oo,      _.push, _.read, _.rnd,_same,_.str

-- `THE` settings is parsed from any `help` i
-- string lines that contain two dashes&nbsp;"`--`".
local THE={}
help:gsub(" [-][-]([^%s]+)[^\n]*%s([^%s]+)",function(key,x) THE[key] = read(x) end)
-- ### Classes

-- (1) ROWS use COLS to make either NUMs or SYMs.
-- (2) ROWS holds data in ROWs, and summarizes columns in NUMs and SYMs.
-- (3) NUMs use SOMEs to store at most `THE.Few` samples per numeric columns.
-- (4) RANGE objects track what `y` values are seen between `xlo` and `xhi`.
local ROWS, COLS, NUM, SYM = klass"ROWS",  klass"COLS",  klass"NUM", klass"SYM"
local ROW = klass"ROW"
local SOME = klass"SOME"
local RANGE = klass"RANGE"
--------------------------------------------------------------------------------
-- ## class RANGE
function RANGE.new(i, xlo, xhi, ys) i.xlo,i.xhi,i.ys,i.rows = xlo,xhi,ys,{} end
function RANGE.add(i,x,y)
  if x < i.xlo then i.xlo = x end -- works for string or num
  if x > i.xhi then i.xhi = x end -- works for string or num
  i.ys:add(y) end

function RANGE.__tostring(i)
  local x, lo, hi = i.ys.txt, i.xlo, i.xhi
  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
--------------------------------------------------------------------------------
-- ## class SOME
function SOME.new(i) i.n,i.t,i.ok=0,{},true end
function SOME.has(i) i.t=i.ok and i.t or sort(i.t); i.ok=true; return i.t end
function SOME.add(i,x)
  if x=="?" then return x end
  i.n=i.n+1
  if      #i.t   < THE.some     then i.ok=false; push(i.t,x)
  elseif rand() < THE.some/i.n then i.ok=false; i.t[rand(#i.t)]=x end end
--------------------------------------------------------------------------------
-- ## class NUM
function NUM.new(i) i.n,i.mu,i.m2,i.w,i.lo,i.hi,i.some=0,0,0,1,big,-big,SOME() end
function NUM.mid(i,p)      return rnd(i.mu,p) end
function NUM.like(i,x,...) return normpdf(x, i.mu, i.sd) end
function NUM.bin(x)
  b=(i.hi - i.lo)/THE.bins; return i.lo==i.hi and 1 or math.floor(x/b+.5)*b end

function NUM.add(i_NUM, v_number)
  if v=="?" then return v end
  i.some:add(v)
  i.n  = i.n + 1
  local d    = v - i.mu
  i.mu = i.mu + d/i.n
  i.m2 = i.m2 + d*(v - i.mu)
  i.sd = i.n<2 and 0 or (i.m2/(i.n-1))^0.5
  i.lo = math.min(v, i.lo)
  i.hi = math.max(v, i.hi) end

function NUM.merge(i,j,      k)
  local k = NUM(i.at, i.txt)
  for _,n in pairs(i.some.t) do k:add(x) end
  for _,n in pairs(j.some.t) do k:add(x) end
  return k end
--------------------------------------------------------------------------------
-- ## class SYM
function SYM.new(i)          i.n,i.syms,i.most,i.mode = 0,{},0,nil end
function SYM.mid(i,...)      return i.mode end
function SYM.like(i,x,prior) return ((i.syms[x] or 0)+THE.m*prior)/(i.n+THE.m) end
function SYM.bin(x)          return x end
function SYM.add(i,v,inc)
  if v=="?" then return v end
  inc=inc or 1
  i.n = i.n + inc
  i.syms[v] = inc + (i.syms[v] or 0)
  if i.syms[v] > i.most then i.most,i.mode = i.syms[v],v end end

function SYM.merge(i,j,      k)
  local k = SYM(i.at, i.txt)
  for x,n in pairs(i.has) do k:add(x,n) end
  for x,n in pairs(j.has) do k:add(x,n) end
  return k end
--------------------------------------------------------------------------------
-- ## class COLS
local  is={}
function is.use(x)     return not x:find":$" end
function is.num(x)     return x:find"^[A-Z]" end
function is.goal(x)    return x:find"[!+-]$" end
function is.klass(x)   return x:find"!$"     end
function is.dislike(x) return x:find"-$"     end

function COLS.new(i,t,     new,is)
  i.xs, i.ys, i.names = {},{},{},t
  for at,txt in pairs(t) do
    new = (is.nump(txt) and NUM or SYM)(at,txt)
    new.usep,  new.w = is.use(txt), is.dislike(txt) and -1 or 1
    if new.usep  then
      if is.klass(new.txt) then i.klass=new end
      push(is.goal(new.txt) and i.ys or i.xs, new) end end end

function COLS.add(i,t)
  for _,cols in pairs{i.xs,i.ys} do
    for _,col in pairs(cols) do col:add(t.cells[col.at]) end end
  return t end
--------------------------------------------------------------------------------
-- ## class ROW
function ROW.new(i,of,t) i.of,i.cells,i.evaled  = of,t,false end
function ROW.klass(i)    return i.cells[i.of.cols.klass.at] end
function ROW.within(i,range)
   local lo, hi, at = range.xlo, range.xhi, range.ys.at
   local v = i.cells[at]
   return  v=="?" or (lo==hi and v==lo) or (lo<v and v<=hi) end
--------------------------------------------------------------------------------
-- ## class ROWS

-- __ROWS(  `t`  :[string]  )__<br>constructor.
function ROWS.new(i,t) i.cols=COLS(t); i.rows={} end
-- __.add(  `t`  :(table|ROW)  )  :ROW__<br>update with a table or ROW.
function ROWS.add(i,t) return push(i.rows, i.cols:add(t.cells and t or ROW(i,t))) end
-- __.mid(  `cols`  :[NUM|SYM],  `p`=2  )  :table__
-- returns `mid`s of some columns; round numerics to `p` decimal places.
function ROWS.mid(i, cols, p,     t)
  t={};for _,col in pairs(cols or i.cols.ys) do t[col.txt]=col:mid(p) end;return t end
-- __.clone(  ?`data`  :(table|[ROW])  )  :ROWS__<br>copy this structure, maybe add data.
function ROWS.clone(i,data,  j)
  j= ROWS(i.cols.names);for _,row in pairs(data or {}) do j:add(row) end; return j end
-- __.like(  `row`  :ROWS,  `nklasses`  :int;  `nrows`  :int  )  :number__
-- how likely is it that `row` could live here?
function ROWS.like(i,row, nklasses, nrows,    prior,like,inc,x)
  prior = (#i.rows + THE.k) / (nrows + THE.k * nklasses)
  like  = math.log(prior)
  for _,col in pairs(i.cols.xs) do
    x = row.cells[col.at]
    if x and x ~= "?" then
      inc  = col:like(x,prior)
      like = like + math.log(inc) end end
  return like end

-- __doRows(  ?`src`  :(string|table),  `fun`  :function( table|ROW )   )__
-- helper function for reading from tables or files. Case arg1 of ...
-- ... _table_  : call  function for all items in table.
-- ... _string_ :  call function on all rows from a file.
-- ... _nil_    :  call function of all rows from standard input.
local function doRows(src, fun)
  if type(src)=="table" then for  _,t in pairs(src) do fun(t) end
                          else for   t in csv(src)   do fun(t) end end end

--------------------------------------------------------------------------------
-- ## class NB
-- (0) Use row1 to initial our `overall` knowledge of all rows.
-- After that (1) add row to `overall` and (2) ROWS about this row’s klass.
-- (3) After `wait` rows, classify row BEFORE updating training knowledge
function NB.new(i,src,report,             row)
  report = report or print
  i.overall, i.dict, i.list  = nil, {}, {}
  doRows(src,  function(row,   k)
    if not i.overall then i.overall = ROWS(row) else  -- (0) eat row1
      row = i.overall:add(row)                    -- add to overall
      if #i.overall.rows > THE.wait then report(row:klass(), i:guess(row)) end
      i:train(row) end end) end                   -- add tp rows’s klass

function NB.train(i,row,      k)
  k=row:klass()
  i.dict[k] = i.dict[k] or push(i.list,i.overall:clone()) -- klass is known
  i.dict[k].txt = k                                      -- each klass knows its name
  i.dict[k]:add(row) end                                 -- update klass with row

function NB.guess(i,row)
    return argmax(i.dict,
      function(klass) return klass:like(row,#i.list,#i.overall.rows) end) end
-- function TREE.new(i,listOfRows,gaurd)
--   i.gaurd, i.kids = gaurd, {}
--   of   = listOfRows[1][1].of
--   best = sort(map(of.cols.x,
--                function(col) i:bins(col,listOfRows) end),lt"div")[1]
--   i.kids = map(best.ranges, function(range)
--               listOfRows1 = {}
    -- local function within(row)       return row:within(best) end
    -- local function withins(rows)     return map(rows, within) end
    -- map(listOrRanges, function(rows) return withins(rows) end) end
    --   tmp= map(rows,withins)
    --   if #tmp > stop then
    --    end)
    --
--------------------------------------------------------------------------------
-- ## class TREE
-- function decisionTree(listOfRows)
-- -- function tree(rows, xols, yklass,y, gaurd)
-- --   local function xranges(xcol) return i:ranges(rows,xcol,yklass,y) end
-- --   i.gaurd = gaurd
-- --   ranges = sort(map(xcols, xranges),lt"div")[1].ranges
-- --   for _,row in pairs(rows) do
-- --     for _,range in pairs(ranges) do
-- --       if row:within(range) then push(range.rows,row) end; break end end
-- --   i.kids = map(ranges,
-- --                  function(range) return TREE(range.rows,xcols,yklass,y,range) end) end
-- --
--   labels , all, xcols = {},{}
--   for label,rows in pairs(listofRows) do
--     for _,row in pairs(rows) do
--       xcols = row.of.cols.xs
--       labels[ push(all,row).id ] = label end end
--   return TREE(all, xcols, SYM, function(row) return labels[row.id] end) end
--
local _ranges, _merge
function _ranges(i,rows,xcol,yklass,y)
  local n,list, dict = 0,{}, {}
  for _,row in pairs(rows) do
    local v = row.cells[xcol.at]
    if v ~= "?" then
      n = n + 1
      local pos = xcol:bin(v)
      dict[pos] = dict[pos] or push(list, RANGE(v,v, yklass(xcol.at, xcol.txt)))
      dict[pos]:add(v, y(row)) end end
  list = sort(list, lt"xlo")
  list = xcol.is=="NUM" and _merge(list, n^THE.min) or list
  return {ranges  = list,
          div   = sum(list,function(z) return z.ys:div()*z.ys.n/n end)} end
```

```

function _merge(b4,min)
  local j,t a,b,c,ay,by,cy = 1,{}
  while j <= #b4 do
    a, b = b4[j], b4[j+1]
    if b then
      ay,by,cy = a.ys, b.ys, a.ys:merge(b.ys)
      if ay.n<min or by.n<min or cy:div() <= (ay.n*ay:div()+by.n*by:div())/cy.n
      then a = raNGE(a.xlo, b.xhi, cy)
           j = j + 1 end end -- skip one, since it has just been merged
    t[#t+1] = a
    j = j + 1 end
  if #t < #b4 then return _merge(t,min) end
  for j=2,#t do t[j].xlo = t[j-1].xhi end
  t[1].xlo, t[#t].xhi = -big, big
  return t end
--------------------------------------------------------------------------------
-- ## TESTS
local no,go = {},{}
function go.the()  print(1); print(THE); return type(THE.p)=="number" and THE.p==2 end

function go.argmax(  t,fun)
  fun=function(x) return -x end
  t={50,40,0,40,50}
  return 3 == argmax(t,fun) end

function go.num(n) n=NUM(); for x=1,100 do n:add(x) end; return n.mu==50.5 end

function go.sym(s)
  s=SYM(); for _,x in pairs{"a","a","a","a","b","b","c"} do s:add(x) end
  return s.mode=="a" end

function go.csv(    n,s)
  n,s=0,0; for row in csv(THE.file)  do n=n+1; if n>1 then s=s+row[1] end end
  return rnd(s/n,3) == 5.441  end

function go.rows(    rows)
  doRows(THE.file,function(t) if rows then rows:add(t)  else rows=ROWS(t) end end)
  return rnd(rows.cols.ys[1].sd,0)==847 end

function go.nb()
  return 268 == #NB("../../data/diabetes.csv").dict["positive"].rows  end

local function _classify(file)
  local Abcd=require"abcd"
  local abcd=Abcd()
  NB(file, function(got,want) abcd:add(got,want) end)
  abcd:pretty(abcd:report())
  return true end

function go.soybean() return _classify("../../data/soybean.csv") end
function go.diabetes() return _classify("../../data/diabetes.csv") end
--------------------------------------------------------------------------------
-- ## START
if    pcall(debug.getlocal, 4, 1)
then  return {ROW=ROW, ROWS=ROWS, NUM=NUM, SYM=SYM, THE=THE,lib=lib}
else  THE = cli(THE,help)
      demos(THE,go) end

--        .---------.
--        |         |
--      -= _________ =-
--         ___   ___
--        |   )=(   |
--         ---   ---
--            ###
--          #  =  #    "This ain't chemistry.
--          #######     This is art."
```