```lua
local b4={}; for k,_ in pairs(_ENV) do b4[k]=k end
local add,big,col,csv,fyi,id,is,klass,lt,map,oo
local per,push, rand, ranges,read, result, seed, splice, str
local help=[[
SAMPLE: while not end of time, look around, see what’s what
(c) 2022 Tim Menzies, timm@ieee.org, BSD2 license

INSTALL: requires: lua 5.4+
         download: sample.lua
         test    : lua sample.lua -h

USAGE: lua sample.lua [OPTIONS]
                                              defaults
                                              ~~~~~~~~
  -S  --Seed  random number seed              = 10019
  -H  --How   optimize for (helps,hurts,tabu) = helps
  -b  --bins  number of bins                  = 16
  -m  --min   min1 size (for pass1)           = .5
  -M  --Min   min2 size (for pass2)           = 10
  -p  --p     distance coefficient            = 2
  -s  --some  sample size                     = 512

OPTIONS (other):
  -f  --file  csv file with data = ../../etc/data/auto93.csv
  -g  --go    start up action    = nothing
  -v  --verbose show details     = false
  -h  --help  show help          = false]]

function read(str)
  str = str:match"^%s*(.-)%s*$"
  if str=="true" then return true elseif str=="false" then return false end
  return math.tointeger(str) or tonumber(str) or str  end

local THE, backup = {}, {}
help:gsub(" [-][-]([^%s]+)[^\n]*%s([^%s]+)",function(key,x)
  for n,flag in ipairs(arg) do
    if flag==("-"..key:sub(1,1)) or flag==("--"..key) then
       x= x=="false" and"true" or x=="true" and"false" or arg[n+1] end end
  x = read(x)
  backup[key] = x
  THE[key] = x end)

if THE.help then os.exit(print(help:gsub("[%u][%u%d]+","\27[1;31m%1\27[0m"))) end

-------------------------------------------------------------------------------
function str(i,        j)
  if type(i)~="table" then return tostring(i) end
  if #i> 0            then return table.concat(map(i,tostring),",") end
  j={}; for k,v in pairs(i) do j[1+#j] = string.format(":%s %s",k,v) end
  table.sort(j)
  return (i.is or "").."{"..table.concat(j," ").."}" end

local _id=0
function is(name,    t)
  local function new(kl,...)
    _id = _id+1
    local x=setmetatable({id=_id},kl); kl.new(x,...); return x end
  t = {__tostring=str, is=name}; t.__index=t
  return setmetatable(t, {__call=new}) end

local ROW,ROWS,SYM,NUM,SOME = is"ROW",is"ROWS",is"SYM",is"NUM",is"SOME"
-------------------------------------------------------------------------------
function col(i,holds,at,txt)
  i.n, i.at, i.txt = 0, at or 0, txt or ""
  i.w= i.txt:find"-$" and -1 or 1
  i.holds = holds end

function add(i,x,inc,fun)
  if x ~= "?" then
    inc = inc or 1
    i.n = i.n + inc
    fun() end
  return  end

function SOME.new(i, ...) col(i,{},...); i.ok=false; end
function SOME.sorted(i,  a)
  if not i.ok then table.sort(i.holds) end; i.ok=true; return i.holds end
function SOME.add(i,x)
  return add(i,x,1,function(     a)
    a = i.holds
    if     #a     < THE.some     then i.ok=false; push(a,x)
    elseif rand() < THE.some/i.n then i.ok=false; a[rand(#a)]=x end end) end

-------------------------------------------------------------------------------
function NUM.new(i, ...) col(i,SOME(),...); i.mu,i.lo,i.hi=0,big,-big end
function NUM.clone(i)    return NUM(i.at, i.txt) end
function NUM.add(i,x)
  return add(i,x,1,function(     d)
    i.holds:add(x)
    d = x - i.mu
    i.mu = i.mu + d/i.n
    i.hi = math.max(x, i.hi); i.lo=math.min(x, i.lo) end ) end

function NUM.merge(i,j,      k)
  local k = NUM(i.at, i.txt)
  for _,x in pairs(i.holds.holds) do k:add(x) end
  for _,x in pairs(j.holds.holds) do k:add(x) end
  return k end

function NUM.mid(i) return i.mu end
function NUM.div(i, a) a=i.holds:all(); return (per(a, .9) - per(a, .1))/2.56 end

function NUM.bin(i,x,   b)
  b = (col.hi - col.lo)/THE.bins; return math.floor(v/b+.5)*b end

-------------------------------------------------------------------------------
function SYM.new( i, ...) col(i,{},...);     i.most, i.mode=0,nil end
function SYM.clone(i) return SYM(i.at, i.txt) end
function SYM.add(i,x,inc)
  return add(i,x,inc, function()
    i.holds[x] = (inc or 1) + (i.holds[x] or 0)
    if i.holds[x] > i.most then i.most,i.mode = i.holds[x],x end end) end

function SYM.merged(i,j,      k)
  local k = SYM(i.at, i.txt)
  for x,n in pairs(i) do k:add(x,n) end
  for x,n in pairs(j) do k:add(x,n) end
  return k end

function SYM.mid(i) return i.mode end
function SYM.div()
  e=0;for k,n in pairs(i.holds) do if n>0 then e=e-n/i.n*math.log(n/i.n,2)end end
  return e end

function SYM.bin(i,x) return x end

function SYM.score(i,want, wants,donts)
  local b, r, z, how = 0, 0, 1/big, {}
  how.helps= function(b,r) return (b<r or b+r < .05) and 0 or b^2/(b+r) end
  how.hurts= function(b,r) return (r<b or b+r < .05) and 0 or r^2/(b+r) end
  how.tabu = function(b,r) return 1/(b+r+z) end
  for v,n in pairs(i.ys.all) do if v==want then b = b+n else r=r+n end end
  return how[the.How](b/(wants+z), r/(donts+z)) end

-------------------------------------------------------------------------------
function ROW.new(i,of,cells) i.of,i.cells,i.evaluated = of,cells,false end
function ROW.__lt(i,j,        n,s1,s2,v1,v2)
  i.evaluated = true
  j.evaluated = true
  s1, s2, n = 0, 0, #i.of.ys
  for _,col in pairs(i.of.ys) do
   v1,v2 = col:norm(i.cells[col.at]), col:norm(j.cells[col.at])
   s1    = s1 - 2.7183^(col.w * (v1 - v2) / n)
   s2    = s2 - 2.7183^(col.w * (v2 - v1) / n) end
  return s1/n < s2/n end

function ROW.within(i,range,         lo,hi,at,v)
   lo, hi, at = range.xlo, range.xhi, range.ys.at
   v = i.cells[at]
   return  v=="?" or lo==hi and v==lo or lo<=v and v<hi end
 -------------------------------------------------------------------------------
function ROWS.new(i,src)
  i.all={}; i.cols={}; i.xs={}; i.ys={}; i.names={}
  if type(src)=="string" then for   row in csv(  src) do i:add(row) end
                         else for _,row in pairs(src) do i:add(row) end end end

function ROWS.clone(i,with,    j)
  j=ROWS({i.names}); for _,r in pairs(with or {}) do j:add(r) end; return j end

function ROWS.add(i,row)
  local function header(   col)
    i.names = row
    for at,s in pairs(row) do
      col = push(i.cols, (s:find"^[A-Z]" and NUM or SYM)(at,s))
      if not s:find":$" then
        if s:find"!$" then i.klass = col end
        push(s:find"[!+-]$" and i.ys or i.xs, col) end end
  end -------------------------------
  if #i.cols==0 then header(row) else
    row = push(i.all, row.cells and row or ROW(i,row))
    for _,col in pairs(i.cols) do col:add(row.cells[col.at]) end end end

function ROWS.bestRest(i,  n,m)
  table.sort(i.all)
  n = #i.all
  m = n^the.min
  return splice(i.all, 1,  m), splice(i.all, n - m) end

function ROWS.mid(i,    p,t)
  t={}; for _,col in pairs(i.ys) do t[col.txt]=col:mid(p) end; return t end

function ROWS.splits(i,bests0,rests0)
  most,range,range1,score = -1
  for _,col in pairs(i.xs) do
    for _,range0 in ranges(col,bests0,rests0) do
      score = range0:score(1,#bests0,#rests0)
      if score>most then most,range1 = score,range0 end end end
  local bests1, rests1 = {},{}
  for _,rows in pairs{bests0,rests0} do
    for _,row in pairs(rows) do
      push(row:within(range1) and bests1 or rests1, row) end end
  return bests1, rests1, range1 end

function ROWS.contrast(i,bests0,rests0,    hows,stop)
  stop = stop or #bests0/4
  hows = hows or {}
  bests1, rests1,range = i:splits(bests0,rests0)
  if (#bests0 + #rests0) > stop and (#bests1 < #bests0 or #rests1 < #rests0) then
    push(hows,range)
    return i:contrast(bests1, rests1, hows, stop) end
  return hows0,bests0 end

-------------------------------------------------------------------------------
function ranges(col, ...)
  local function xpand(t)
    for j=2,#t do t[j].xlo = t[j-1].xhi end
    t[1].xlo, t[#tmp].xhi = -big, big
    return t end
  local function merged(i,j,min,      k)
    k = i:merge(j)
    if i.n < min or j.n < min or k:div()<=(i.n*i:div() + j.n*j:div())/k.n then
      return k end end
  local function merge(b4,min,      t,j,a,b,c)
    t,j = {},1
    while j <= #b4 do
      a, b = b4[j], b4[j+1]
      if b then
         c = merged(a.ys, b.ys, min)
         if c then
           j = j + 1
           a = {xlo=a.xlo, xhi=b.xhi, ys=c} end end
      t[#t+1] = a
      j = j + 1 end
    return #b4 == #t and t or merge(t,min)
  end ---------------------------
  local known,out,n,v,x = {},{}, 0
  for klass,rows in pairs{...} do
    n = n + #rows
    for _,row in pairs(rows) do
      v = row.cells[col.at]
      if v ~= "?" then
        x = col:bin(v)
        known[x] = known[x] or push(out,{xlo=v, xhi=v, ys=col:clone()})
        if v < known[x].xlo then known[x].xlo = v end -- works for string or num
        if v > known[x].xhi then known[x].xhi = v end -- works for string or num
        known[x].ys:add(klass) end end end
  table.sort(out,lt("xlo"))
  out= col.is=="NUM" and xpand(merge(out, n^THE.bins)) or out
  return #out < 2 and {} or out end
```

```
--------------------------------------------------------------------------------
oo  = function(i) print(str(i)) end
big = math.huge
fyi = function(...) if THE.verbose then print(...) end end
fmt = table.format
rand= math.random

function push(t,x)    t[1+#t]=x; return x end
function map(t,f,  u) u={}; for k,v in pairs(t) do u[1+#u]=f(v) end return u end
function per(t,p)     p=p*#t//1; return t[math.max(1,math.min(#t,p))] end
function lt()         return function(a,b) return a[x] < b[x] end end

function splice( t, i, j, k,    u)
  u={}; for n=(i or 1)//1, (j or #t)//1, (k or 1)//1 do u[1+#u]=t[n] end return u
end

function csv(csvfile)
  csvfile = io.input(csvfile)
  return function(s, t)
    s=io.read()
    if not s then io.close(csvfile) else
      t={}; for x in s:gmatch("([^,]+)") do t[1+#t] = read(x) end
      return t end end end
--------------------------------------------------------------------------------
local fails,go,no=0,{},{}

function go.the() fyi(str(THE));  str(THE) return true end

function go.some( s)
  THE.some = 16
  s=SOME(); for i=1,10000 do s:add(i) end; oo(s:all())
  oo(s:all())
  return true end

function go.num( n)
  n=NUM(); for i=1,10000 do n:add(i) end; oo(n)
  return true end

function go.sym( s)
  s=SYM(); for i=1,10000 do s:add(math.random(10)) end;
  return s.holds[9]==1045  end

function go.csv()
  for row in csv(THE.file) do oo(row) end; return true; end

function go.rows( rows)
  rows = ROWS(THE.file);
  map(rows.ys,print); return true; end

function go.mid(  r)
  r= ROWS(THE.file)

end
--------------------------------------------------------------------------------
local going={}
for s,_ in pairs(go) do going[1+#going]=s end
table.sort(going)

for _,s in pairs(go[THE.go] and {THE.go} or going) do
  for k,v in pairs(backup) do THE[k]=v end
  math.randomseed(THE.Seed)
  io.write(".")
  result = go[s]()
  if result ~= true then
    fails = fails + 1
    print("--Error",s,status) end end

for k,v in pairs(_ENV) do if not b4[k] then print("?",k,type(v)) end end
os.exit(fails)
```