```
                                    __
                                   /\_ \
   ____     __      ___ ___        \//\ \    __  __     __
  /',__\  /'__`\  /' __` __`\        \ \ \  /\ \/\ \  /'__`\
 /\__, `\/\ \L\.\_/\ \/\ \/\ \  __    \_\ \_\ \ \_\ \/\ \L\.\_
 \/\____/\ \__/.\_\ \_\ \_\ \_\/\_\   /\____\\ \____/\ \__/.\_\
  \/___/  \/__/\/_/\/_/\/_/\/_/\/_/   \/____/ \/___/  \/__/\/_/


-- sam.lua : reasoning via minimal sampling arcoss the data
-- (c)2022 Tim Menzies <timm@ieee.org> BSD 2 clause license
local l=require"lib"
local any,cat,cli,coerce,copy,csv = l.any,l.cat,l.cli,l.coerce,l.copy,l.csv
local lines,many,obj,per,push = l.lines,l.many,l.obj,l.per,l.push
local rogues,words = l.rogues,l.words

local rand = math.random
local Cols,Data,Row,Num,Sym = obj"Cols", obj"Data", obj"Row",obj"Num", obj"Sym"

local the={bins    = 10,
           cohen   = .35,
           example = "ls",
           ratios  = 256,
           seed    = 10019,
           some    = 512}

-- Num -------------------------------------------------------------------------
function Num:new(at,txt)
  txt = txt or ""
  return {n=0,at=at or 0, txt=txt, details=nil, has={},
          hi= -math.huge, lo= math.huge, w=txt:find"-$" and -1 or 1} end

function Num:add(x)
  if x ~= "?" then
    local pos
    self.n  = self.n + 1
    self.lo = math.min(x, self.lo)
    self.hi = math.max(x, self.hi)
    if     #self.has < the.ratios        then pos = 1 + (#self.has)
    elseif rand()    < the.ratios/self.n then pos = rand(#self.has) end
    if pos then self.details  = nil
                self.has[pos] = x end end end

function Num:bin(x,     a,b,lo,hi)
  local b = (self.hi - self.lo)/the.bins
  return self.hi==self.lo and 1 or math.floor(x/b+.5)*b  end

function Num:discretize(x)
  _, details = self:holds()
  for _,bin in pairs(details) do
    if x> bin.lo and x<=bin.hi then return x end end 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 function _div(a,epsilon,bins,   inc,one,all)
  inc = #a // bins
  one = {lo=a[1], hi=a[1], n=0}
  all = {one}
  for i = 1,#a-inc do
    if   one.n >= inc and a[i] ~= a[i+1] and one.hi-one.lo > epsilon
    then one = push(all, {lo=one.hi, hi=a[i], n=0}) end
    one.hi  = a[i]
    one.n   = bin.n + 1 end
  all[1].lo     = -math.huge
  all[#bins].hi = math.huge
  return all end

function Num:holds(    inc,i)
  if not self.details then
    table.sort(self.has)
   self.details = _div(self.has, self:div()*my.cohen, my.bins); end
  return self.has, self.details end

function Num:mid() return per(self:holds(),.5) end

function Num:norm(num)
  return self.hi - self.lo < 1E-9 and 0 or (num-self.lo)/(self.hi-self.lo) end

function Num:div(  a)
  a=self:holds()
  return (per(a,.9) - per(a,.1))/2.58 end

-- Sym -------------------------------------------------------------------------
function Sym:new(at,txt)
  return {n=0,at=at or 0, txt=txt or "", ready=false, has={}} end

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

function Sym:discretize(x) return x end

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

function Sym:mid(    mode,most)
  for k,n in pairs(i.has) do if not mode or n>most then mode,most=k,n end end
  return mode end

function Sym:div(  e)
  local function p(x) return x*math.log(x,2) end
  e=0; for _,v in pairs(i.has) do if v>0 then e=e-p(v/i.n) end; return e end end

-- Row ------------------------------
-- function Data.far(XXX) end
--

function Data:half(rows, above, all)
   local all  = all or self.rows
   local some = many(all, the.some)
   local left = above or far(any(some), some) end
-- (defmethod half ((i rows) &optional all above)
--   "Split rows in two by their distance to two remove points."
--   (let* ((all   (or    all (? i _has)))
--          (some  (many  all (! my some)))
--          (left  (or    above (far (any some) some)))
--          (right (far   left some))
--          (c     (dists left right))
--          (n 0)  lefts rights)
--     (labels ((project (row)
--                 (let ((a (dists row left))
--                       (b (dists row right)))
--                   (cons (/ (+ (* a a) (* c c) (- (* b b))) (* 2 c)) row))))
--       (dolist (one (sort (mapcar #'project all) #'car<))
--         (if (<= (incf n) (/ (length all) 2))
--           (push (cdr one) lefts)
--           (push (cdr one) rights)))
--       (values left right lefts rights c))))
--
-- -----------------------------------------------------------------------------
return {the=the,Cols=Cols, Data=Data, Num=Num, Sym=Sym}
. .
|*|_
||[_)

-- lib.lua : some of my favorite lua tricks.
-- (c)2022 Tim Menzies <timm@ieee.org> BSD 2 clause license
local l={}

-- Cache names -----------------------------------------------------------------
local b4={}; for k,_ in pairs(_ENV) do b4[k]=k end
function l.rogues()
  for k,v in pairs(_ENV) do if not b4[k] then print("?",k,type(v)) end end end

-- Print table -----------------------------------------------------------------
function l.chat(t) print(l.cat(t)); return t end

function l.cat(t)
  if type(t)~="table" then return tostring(t) end
  local function show(k,v)
    if not tostring(k):find"^[A-Z]"  then
      v=l.cat(v)
      return #t==0 and string.format(":%s %s",k,v) or tostring(v) end end
  local u={}; for k,v in pairs(t) do u[1+#u] = show(k,v) end
  table.sort(u)
  return (t._is or "").."{"..table.concat(u," ").."}" end

-- Maths ----------------------------------------------------------------------
function l.rnd(num, places)
  local mult = 10^(places or 3)
  return math.floor(num * mult + 0.5) / mult end

-- Lists -----------------------------------------------------------------------
function l.any(t) return t[math.random(#t)] end

function l.copy(t)
  if type(t) ~= "table" then return t end
  local u={}; for k,v in pairs(t) do u[k] = l.copy(v) end
  return setmetatable(u,getmetatable(t))  end

function l.least(t,x,    y)
  for _,n in pairs(t) do y=n; if x <= y then break end end
  return y end

function l.many(t,n,  u)  u={}; for i=1,n do u[1+#u]=l.any(t) end; return u end

function l.per(t,p)
  p=p or .5
  p=math.floor((p*#t)+.5); return t[math.max(1,math.min(#t,p))] end

function l.push(t,x) t[1+#t]=x; return x end

-- Update slots in `t` from command line ---------------------------------------
function l.cli(t)
  for slot,v in pairs(t) do
    v = tostring(v)
    for n,x in ipairs(arg) do
      if x=="-"..(slot:sub(1,1)) or x=="--"..slot then
        v = v=="false" and "true" or v=="true" and "false" or arg[n+1] end end
    t[slot] =  l.coerce(v) end
  return t end

-- Define classes --------------------------------------------------------------
function l.obj(name)
  local function new(k,...)
    local self = setmetatable({},k)
    return setmetatable(k.new(self,...) or self,k) end
  local t={_is = name, __tostring = l.cat}
  t.__index = t
  return setmetatable(t,{__call=new}) end

-- Coerce ----------------------------------------------------------------------
function l.coerce(str)
  local function coerce1(str)
    if str=="true"  then return true end
    if str=="false" then return false end
    return str end
  return tonumber(str) or coerce1(str:match"^%s*(.-)%s*$") end

-- Coerce lines from csv file (fiterling result through `fun`).
function l.csv(filename, fun)
  l.lines(filename, function(t) fun(l.words(t,",",l.coerce)) end) end

--- Call `fun  on all lines from `filename`.
function l.lines(filename, fun)
  local src = io.input(filename)
  while true do
    local str = io.read()
    if not str then return io.close(src) else fun(str) end end end

-- Split  `str` on `sep`, filtering parts through `fun`.
function l.words(str,sep,fun,      t)
  fun = fun or function(z) return z end
  sep = l.string.format("([^%s]+)",sep)
  t={};for x in str:gmatch(sep) do t[1+#t]=fun(x) end;return t end

return l

 _  _
(/,(_]
   ._|
-- eg.lua : demo code for sam.lua
-- (c)2022 Tim Menzies <timm@ieee.org> BSD 2 clause license
local l=require"lib"
local _=require"sam"
local cat,chat,cli,copy,per = l.cat,l.chat,l.cli,l.copy,l.per
local rogues = l.rnd,l.rogues
local Num = _.Num
local the,eg,fails = _.the,{},0

local function run(k,    b4,out)
  math.randomseed(the.seed)
  b4=copy(the); out=eg[k].fun(); the=copy(b4); return out==true end

local function egs(   t)
  t={}; for k,v in pairs(eg) do t[1+#t]=k end; table.sort(t);  return t end

eg.the = {doc="show config", fun=function ()
  chat(the); return true end}

eg.ls = {doc="list examples", fun=function ()
  print("\nExamples (lua eg.lua -f X):\nX=")
  for _,k in pairs(egs()) do print(string.format("%7s : %s",k,eg[k].doc)) end
  return true end}

eg.all = {doc="run all examples", fun=function()
  for _,k in pairs(egs()) do
    if k ~= "all" then
      if not run(k) then fails = fails + 1; print("FAIL!",k) end end end end}

eg.nums = {doc="numbers", fun=function(   n)
  n=Num()
  the.keep = 64
  for i=1,100 do n:add(i) end
  return 52==n:mid(r) and 32.56==rnd(n:div(),2)  end}

the = cli(the)
if eg[the.example] then eg[the.example].fun() end
rogues()
os.exit(fails)
```