```lua
local b4={}; for k,_ in pairs(_ENV) do b4[k]=k end
local the,help={},[[

lua l5.lua [OPTIONS]
L5 == a very little LUA learning lab
(c)2022, Tim Menzies, BSD 2-clause license

OPTIONS (for changing the inference):

  -cohen  -c  F  cohen's small effect size      = .35
  -far    -F  F  look no further than "far"     = .9
  -keep   -k     items to keep in a number      = 512
  -leaves -l     leaf size                      = .5
  -p      -p  P  distance calcs coefficient     = 2
  -seed   -S  P  random number seed             = 10019
  -some   -s     look only at "some" items      = 512

OPTIONS (for housekeeping):

  -dump   -d     exit on error, with stacktrace = false
  -file   -f  S  where to get data              = ../etc/data/auto93.csv
  -help   -h     show help                      = false
  -rnd    -r  S  format string                  = %5.2f
  -todo   -t  S  start-up action                = nothing


KEY: S=string, P=poisint, F=float
]]

local as = setmetatable
local function obj(   t)
  t={__tostring=o}; t.__index=t
  return as(t, {__call=function(_,...) return t.new(_,...) end}) end
-----------------------------------------------------------------------------
-
---        _       _
---     __| | __ _| |_ __ _
---    / _` |/ _` | __/ _` |
---   | (_| | (_| | || (_| |
---    \__,_|\__,_|\__\__,_|

local Sym, Num = obj(), obj()
function Sym:new(at,s) return as({
  is="Sym",      -- type
  at=at or 0,    -- column index
  name=s or "",  -- column name
  n=0,           -- number of items summarized in this column
  all={},        -- all[x] = n means we've seen "n" repeats of "x"
   most=0,       -- count of the most frequently seen symbol
   mode=nil      -- the most commonly seen letter
  }, Sym) end

function Num:new(at,s) return as({
  is="Num",       -- type
  at=at or 0,     -- column index
  name=s or "",   -- column name
  n=0,            -- number of items summarizes in this column
  mu=0,           -- mean (updated incrementally)
  m2=0,           -- second moment (updated incrementally)
  sd=0,           -- standard deviation
  all={},         -- a sample of items seen so far
  lo=1E31,        -- lowest number seen
  hi=-1E31,       -- highest number seen
  w=(s or ""):find"-$" and -1 or 1 -- "-1"= minimize and "1"= maximize
  }, Num) end

local function Egs(names)  return {
  is="egs",      -- type
  all={},        -- all the rows
  names=names,   -- list of name
  cols={},       -- list of all columns  (Nums or Syms)
  x={},          -- independent columns (nothing marked as "skip")
  y={}           -- dependent columns (nothing marked as "skip")
  } end

--[[
## Coding Conventions
- "i" not "self"
- if something holds a list of thing, name the holding variable "all"
- no inheritance
- only define a method if that is for polymorphism
- when you can, write functions down on one line
- all config items into a global "the" variable
- all the test cases (or demos) are "function Demo.xxx".
- random seed reset so carefully, just once, at the end of the code.
]]
---                _    _  _
---     _   _ | |_ (_)| | ___
---    | | | || __|| || |/ __|
---    | |_| || |_ | || |\__ \
---     \__,_| \__||_||_||___/
-------------------------------------------------------------------------------
local r   = math.random
local fmt = string.format
local function push(t,x) table.insert(t,x); return x end
---    ____ ____ ____ ____ ____ ____
---    |    |  | |___ |__/ |    |___
---    |___ |__| |___ |  \ |___ |___

local thing,things,file2things
function thing(x)
  x = x:match"^%s*(.-)%s*$"
  if x=="true" then return true elseif x=="false" then return false end
  return tonumber(x) or x end

function things(x,sep,  t)
  t={}; for y in x:gmatch(sep or"([^,]+)") do push(t,thing(y)) end
  return t end

function file2things(file,      x)
  file = io.input(file)
  return function()
    x=io.read();
    if x then return things(x) else io.close(file) end end end
---    ____ ____ ___    ____ ____ ___
---    | __ |___  |     [__  |___  |
---    |__] |___  |  .  ___] |___  |
---              ,
local last,per,any,many
function last(a)       return a[ #a ] end
function per(a,p)      return a[ (p*#a)//1 ] end
function any(a)        return a[ math.random(#a) ] end
function many(a,n,  u) u={}; for j=1,n do push(u,any(a)) end; return u end
---    ___ ____ ___  _    ____
---     |  |__| |__] |    |___
---     |  |  | |__] |___ |___

local firsts,sort,map,slots
function firsts(a,b)  return a[1] < b[1] end
function sort(t,f)    table.sort(t,f); return t end
function map(t,f, u)  u={};for k,v in pairs(t) do push(u,f(v)) end; return u end
function slots(t, u,s)
  u={}
  for k,v in pairs(t) do s=tostring(k);if s:sub(1,1)~="_" then push(u,k) end end
  return sort(u) end
---    ___  ____ _ _  _ ___
---    |__] |__/ | |\ |  |
---    |    |  \ | | \|  |

local oo,o, rnd, rnds
function oo(t) print(o(t)) end
function o(t,seen,        key,xseen,u)
  seen = seen or {}
  if type(t)~="table" then return tostring(t) end
  if seen[t]          then return "..." end
  seen[t] = t
  key   = function(k) return fmt(":%s %s",k,o(t[k],seen)) end
  xseen = function(x) return o(x,seen) end
  u = #t>0 and map(t,xseen) or map(slots(t),key)
  return (t.is or "")..'{'..table.concat(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
---    ____ ___ ____ ____ ___     _  _ ___
---    [__   |  |__| |__/  |  __  |  | |__]
---    ___]  |  |  | |  \  |      |__| |

local Demo, ok = {fails=0}
function ok(test,msg)
  print(test and "PASS: "or "FAIL: ",msg or "")
  if not test then
    Demo.fails=Demo.fails+1
    if the.dump then assert(test,msg) end end end

function Demo.main(todo,seed)
   for k,one in pairs(todo=="all" and slots(Demo) or {todo}) do
     if k ~= "main" and type(Demo[one]) == "function" then
       math.randomseed(seed)
       Demo[one]() end end
   for k,v in pairs(_ENV) do if not b4[k] then print("?",k,type(v)) end end
   return Demo.fails end

local 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
      if x=="false" then the[key]=false elseif x=="true" then the[key]=true else
      d[key] = tonumber(x) or x end end)
  if d.help then print(help) end
  return d end
```

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

local add
function add(i,x, inc)
  inc = inc or 1
  if x ~= "?" then
    i.n = i.n + inc
    i:add1(x,inc) end
  return x end

function Sym.add1(i,x,inc)
  i.all[x] = inc + (i.all[x] or 0)
  if i.all[x] > i.most then i.most, i.mode = i.all[x], x end end

function Num.add1(i,x,inc,    d)
  for j=1,inc do
    d     = x - i.mu
    i.mu  = i.mu + d/i.n
    i.m2  = i.m2 + d*(x - i.mu)
    i.sd  = (i.m2<0 or i.n<2) and 0 or ((i.m2/(i.n-1))^0.5)
    i.lo  = math.min(x, i.lo)
    i.hi  = math.max(x, i.hi)
    if     #i.all < the.keep      then push(i.all,x)
    elseif r()    < they.keep/i.n then i.all[r(#i.all)]=x end end end
--- _  _ ____ _  _ ____    ___  ____ ___ ____
--- |\/| |__| |_/  |___    |  \ |__|  |  |__|
--- |  | |  | | \_ |___    |__/ |  |  |  |  |

local header,data,file2Egs
function header(names,   i,col)
  i = Egs(names)
  for at,name in pairs(names) do
    col = push(i.cols, (name:find"^[A-Z]" and Num or Sym)(at,name))
    if not name:find":$" then
      push(name:find"[-+]$" and i.y or i.x, col) end end
  return i end

function data(i,row)
  push(i.all, row)
  for _,col in pairs(i.cols) do add(col, row[col.at]) end
  return i end

function file2Egs(file,   i)
  for row in file2things(file) do
    if i then data(i,row) else i = header(row) end end
  return i end
--- ____ _  _ _  _ _  _ ____ ____ _ ___  ____
--- [__  |  | |\/| |\/| |__| |__/ |   /  |___
--- ___] |__| |  | |  | |  | |  \ |  /__ |___

function Sym.mid(i) return i.mode end
function Sym.div(i,  e)
  e=0; map(i.all,function(n) e = e + n/i.n * math.log(n/i.n,2) end)
  return -e end

function Num.mid(i) return i.mu end
function Num.div(i) return i.sd end

function Num.clone(i) return Num(i.at, i.name) end
function Sym.clone(i) return Sym(i.at, i.name) end

local mids
function mids(cols,rows,    seen,tmp)
  seen = function(col) return col:clone() end
  tmp  = map(cols, seen)
  for _,row in pairs(rows) do
    for _,seen in pairs(tmp) do
      add(seen, row[seen.at]) end end
  return rnds(map(tmp, function(seen) return seen:mid() end)) end
--- ___  _ ____ ___ ____ _  _ ____ ____
--- |  \ | [__   |  |__| |\ | |    |___
--- |__/ | ___]  |  |  | | \| |___ |___

local far,furthest,neighbors,dist
function far(      i,r1,rows,far)
  return per(neighbors(i,r1,rows),far or the.far)[2] end

function furthest( i,r1,rows)
  return last(neighbors(i,r1,rows))[2] end

function neighbors(i,r1,rows)
  return sort(map(rows, function(r2) return {dist(i,r1,r2),r2} end),firsts) end

function dist(i,row1,row2,    d,n,a,b,inc)
  d,n = 0,0
  for _,col in pairs(i.x) do
    a,b = row1[col.at], row2[col.at]
    inc = a=="?" and b=="?" and 1 or col:dist1(a,b)
    d = d + inc^the.p
    n = n + 1 end
  return (d/n)^(1/the.p) end

function Sym.dist1(i,a,b) return a==b and 0 or 1 end

function Num.dist1(i,a,b)
  if     a=="?" then b=i:norm(b); a=b<.5 and 1 or 0
  elseif b=="?" then a=i:norm(a); b=a<.5 and 1 or 0
  else   a,b = i:norm(a), i:norm(b)  end
  return math.abs(a - b) end

function Num.norm(i,x)
  return i.hi - i.lo < 1E-32 and 0 or (x - i.lo)/(i.hi - i.lo) end
--- ____ _    _  _ ____ ___ ____ ____
--- |    |    |  | [__   |  |___ |__/
--- |___ |___ |__| ___]  |  |___ |  \

local half, cluster, clusters
function half(i, rows,    project,row,some,east,west,easts,wests,c,mid)
  function project(row,a,b)
    a= dist(i,east,row)
    b= dist(i,west,row)
    return {(a^2 + c^2 - b^2)/(2*c), row}
  end -------------------------
  some = many(rows, the.some)
  east = furthest(i,any(some), some)
  west = furthest(i,east,         some)
  c    = dist(i,east,west)
  easts,wests = {},{}
  for n, xrow in pairs(sort(map(rows,project),firsts)) do
    row = xrow[2]
    if n==#rows//2 then mid=row end
    push(n <= #rows//2 and easts or wests, row) end
  return easts, wests, east, west, mid   end

function cluster(i,rows,  here,lefts,rights)
  rows = rows or i.all
  here = {all=rows}
  if #rows > 2*(#i.all)^the.leaves then
    lefts, rights = half(i, rows)
    if #lefts < #rows then
      here.lefts = cluster(i,lefts)
      here.rights= cluster(i,rights) end end
  return here end

function clusters(i,t,pre)
  if t then
    pre = pre or ""
    if not t.lefts and not t.rights then
      print(fmt("%5s %-20s",#t.all, pre), o(mids(i.y,t.all)))
    else
      print(fmt("%5s %-20s",#t.all, pre))
      clusters(i,t.lefts,  "|.. ".. pre)
      clusters(i,t.rights, "|.. ".. pre) end end end
--- ___  _ ____ ____ ____ ____ ___ _ ___  ____
--- |  \ | [__  |    |__/ |___  |  |   /  |___
--- |__/ | ___] |___ |  \ |___  |  |  /__ |___
local merge,merged
function Sym.spans(i, j)
  local xys,all,one,last,x,y,n = {}, {}
  for x,n in pairs(i.all) do push(xys, {x,"easts",n}) end
  for x,n in pairs(j.all) do push(xys, {x,"wests",n}) end
  for _,tmp in ipairs(sort(xys,firsts)) do
    x,y,n = unpack(tmp)
    if x ~= last then
      last = x
      one  = push(all, {lo=x, hi=x, all=Num(i.at,i.txt)}) end
    add(one.all, y, n) end
  return all end

function Num.spans(i, j)
  local xys,all,lo,hi,gap,one,x,y,n = {},{}
  lo,hi = math.min(i.lo, j.lo), math.max(i.hi,j.hi)
  gap   = (hi - lo) / (6/the.cohen)
  for _,n in pairs(i.all) do push(xys, {n,"easts",1}) end
  for _,n in pairs(j.all) do push(xys, {n,"wests",1}) end
  one = {lo=lo, hi=lo, all=Sym(i.at,i.txt)}
  all = {one}
  for _,tmp in ipairs(sort(xys,firsts)) do
    x,y,n = unpack(tmp)
    if   one.hi - one.lo > gap
    then one = push(all, {lo=one.hi, hi=x, all=Sym(i.at,i.txt)}) end
    one.hi = x
    add(one.all,y,n) end
  all          = merge(all)
  all[1   ].lo = -math.huge
  all[#all].hi =  math.huge
  return all end

function merge(b4,      j,n,now,a,b,both)
  j, n, now = 0, #b4, {}
  while j < #b4 do
    j    = j+1
    a, b = b4[j], b4[j+1]
    if b then
      both = merged(a,b)
      if both then a, j = {lo=a.lo, hi=b.hi, all=both}, j+1 end end
    push(now,a)
    j = j+1 end
  return #now == #b4 and b4 or merge(now) end

function merged(i,j,    k,ei,ej,ek)
  k = Sym(i.at,i.txt)
  for x,n in pairs(i.all) do add(k,x,n) end
  for x,n in pairs(j.all) do add(k,x,n) end
  ei, ej, ek= div(i), div(j), div(k)
  if i.n==0 or j.n==0 or 1.01*ek <= (i.n*ei + j.n*ej)/(i.n+j.n) then
    return k end end
```

```
--------------------------------------------------------------------------------
function Demo.the() oo(the) end

function Demo.many(a)
  a={1,2,3,4,5,6,7,8,9,10}; ok("{10 2 3}" == o(many(a,3)), "manys") end

function Demo.egs()
  ok(5140==file2Egs(the.file).y[1].hi,"reading") end

function Demo.dist(i)
  i = file2Egs(the.file)
  for n,row in pairs(i.all) do print(n,dist(i, i.all[1], row)) end end

function Demo.far(  i,j,row1,row2,row3,d3,d9)
  i = file2Egs(the.file)
  for j=1,10 do
    row1 = any(i.all)
    row2 = far(i,row1, i.all, .9)
    d9   = dist(i,row1,row2)
    row3 = far(i,row1, i.all, .3)
    d3   = dist(i,row1,row3)
    ok(d3 < d9, "closer far") end end

function Demo.half(  i,easts,wests)
  i = file2Egs(the.file)
  easts,wests = half(i, i.all)
  oo(mids(i.y, easts))
  oo(mids(i.y, wests)) end

function Demo.cluster(   i)
  i = file2Egs(the.file)
  i = file2Egs(the.file)
  clusters(i,cluster(i))
 end

--------------------------------------------------------------------------------
the=settings(help)
Demo.main(the.todo, the.seed)
```