```lua
--------------------------------------------------------------------------------
---     __         ______                                                    ___
---   /\ \       /\  ___\                                                ,o88888
---   \ \ \      \ \ \__/                                              ,o8888888'
---    \ \ \  __  \ \___``\                            ,:o:o:oooo.        ,8O88Pd8888"
---     \ \ \L\ \  \/\ \L\ \                       ,.::.::o:ooooOoOoO. ,oO8O8Pd888'"
---      \ \_____\   \ \_____/                    ,.:.::o:ooOoOoO8O8OOo.8OOPd8O8O"
---       \/____/     \/___/                     , ..:.::o:ooOoOOOO8OOOOo.FdO8O8"
---                                              , ..:.::o:ooOoOO8O888O8O,COCOO"
---   a little LUA learning library              , . ..:.::o:ooOoOOOO8OOOOCOCO"
---   (c) Tim Menzies 2022, BSD-2                 . ..:.::o:ooOoOoOO8O8OCCCC"o
---   https://menzies.us/l5                        . ..:.::o:ooooOoCoCCC"o:o
---   Share and enjoy                              . ..:.::o:o:,cooooCo"oo:o:
---                                            `   . . ..:.:cocoooo"'o:o:::'
---                                            .`   . ..::ccccoc"'o:o:o:::'
---                                           :.:.    ,c:cccc"':.:.:.:.:.'
---                                         ..:.:"'`::::c:"'..:.:.:.:.:.'
---                                       ...:.'.:.::::"'    . . . . .'
---                                      .. . ....:."' `   .  . . ''
---                                     . . . ...."'
---                                     .. . ."'     -hrr-
---                                    .
---
---
--------------------------------------------------------------------------------
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

OPTIONS (inference):                                  | DEFAULT
  -boot   -b P  #bootstrap samples                    | 256
  -cohen  -c F  cohen's small effect size             | .35
  -cliffs -C F  threshold on Cliff's delta            | .147
  -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 (housekeeping):
  -dump   -d    on error, exit+ 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
]]
--------------------------------------------------------------------------------
--[[
Copyright 2022, Tim Menzies

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:

1. Redistributions of source code must retain the above copyright
notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright
notice, this list of conditions and the following disclaimer in the
documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,
BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
POSSIBILITY OF SUCH DAMAGE.  --]]

--------------------------------------------------------------------------------
-- ## Coding Conventions
--
-- - All config options in "the" (which is generated by parsing the help text)
-- - Line width = 80
-- - when you can, write functions down on one line
-- - "i" not "self" (so we can fit more on each line)
-- - if something holds a list of thing, name the holding variable "all"
-- - no inheritance
-- - only define a method if that is for polymorphism
-- - all config items into a global "the" variable
-- - all the test cases (or demos) are "function Demo.xxx".
--   - If test case assertion crashed, add "1" to Demo.fails
--   - On exit return the value of Demo.fails as the exit status
-- - random seed reset so carefully, just once, at the end of the code.
-- - usually, no line with just "end" on it
--------------------------------------------------------------------------------
---     ____  _____  ___  _____
---    |    \ |___|   |   |___|
---    |____/ |   |   |   |   |

-- This code reads date from csv files (where "?" denotes "missing value").
local is={}
function is.missing(x) return x=="?" end

-- The names on  row1 of that file define the role of that column.
-- Names in row1 ending with ":" are to be ignored
function is.skip(x) return x:find":$" end

-- Names in row1 starting in upper case are numbers
function is.num(x) return x:find"^[A-Z]" end

-- Names in row1 ending with "!" are classes.
function is.class(x) return x:find"!$" end

-- Names in row1 ending with "-" are objectives to be minimized.
function is.less(x) return x:find"-$" end

-- Names in row1 ending with "+" are objectives to be maximized.
function is.more(x) return x:find"+$" end

-- Objectives or classes are dependent variables.
function is.dependent(x) return is.more(x) or is.less(x) or is.class(x) end

-- For example, in this data file, we will ignore column 3 (Hp:),
-- try to minimize weight (Lbs-) and maximize acceleration and
-- miles per hour (Acc+, Mpg+). Also, with one exception (origin),
-- everything is numeric. Finally,  there are some missing values on
-- lines 3 and lines 7.
--
--      Clndrs, Weight, Hp:, Lbs-, Acc+, Model, origin, Mpg+
--      8,      304.0,  193, 4732, 18.5, 70,    1,      10
--      8,      ?,      215, 4615, 14,   70,    1,      10
--      4,      85,     70,  2070, 18.6, 78,    3,      40
--      4,      85,     65,  2110, 19.2, 80,    3,      40
--      4,      85,     ?,   1835, 17.3, 80,    2,      40
--      4,      98,     76,  2144, 14.7, 80,    2,      40

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

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 = obj() -- Where to summarize symbols
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

local Num = obj() -- Where to summarize numbers
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
  ok=false,      -- true if "all" is sorted
  all={},        -- a sample of items seen so far
  lo=1E31,       -- lowest number seen; initially, big so 1st num sends it low
  hi=-1E31,      -- highest number seen;initially, msall to 2st num sends it hi
  w=is.less(s or "") and -1 or 1 -- "-1"= minimize and "1"= maximize
  }, Num) end

local Egs = obj() -- Where to store examples, summarized into Syms or Nums
function Egs:new(names,     i,col,here)  i=as({
  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")
  class=nil      --  classes
  },Egs)
  for at,name in pairs(names) do
    col = (is.num(name) and Num or Sym)(at,name)
    i.cols[1+#i.cols] = col
    here = is.dependent(name) and i.y or i.x
    if not is.skip(name) then
      here[1 + #here] = col
      if is.class(name) then i.class=col end end end
  return i end
---      _ |  _   _  .  _   _
---     (_ | (_) | | | | | (_|
---                         _|

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

local data
function Egs.clone(i,rows,    copy)
  copy = Egs(i.names)
  for _,row in pairs(rows or {}) do  data(copy,row)  end
  return copy end
```

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

local r   = math.random
local fmt = string.format
local unpack = table.unpack
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 t[1+#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(txt) end
  return d end
-------------------------------------------------------------------------------
---      _  _ ___ ___    ___ ___ ___ ___ ___
---     |  | [__  |__    |    |__| [__  |__  [__
---     |__| ___] |___   |___ |  | ___] |___ ___]

---           _|  _ _|_ _    _ _ |
---     |_||_)(_|(_| | (/_  (_(_)|_\
---         |

local add
function add(i,x, inc)
  inc = inc or 1
  if not is.missing(x) then
    i.n = i.n + inc
    i:internalAdd(x,inc) end
  return x end

function Sym.internalAdd(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.internalAdd(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 i.ok=false; push(i.all,x)
    elseif r() < they.keep/i.n then i.ok=false; i.all[r(#i.all)]=x end end end

function Num.sorted(i)
  if not i.ok then i.all = sort(i.all) end
  i.ok=true
  return i.all end
---      _ _  _ |  _   _| _ _|_ _
---     | | |(_||<(/_ (_|(_| | (_|

local file2Egs -- not "local data" (since defined above)
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 = Egs(row) end end
  return i end
---      _       _  _    _  _  _    _  ._  _
---     _\|_||‾ |‾ ||‾ |‾ |(_||‾ |/_(/_

local mids
function mids(i,rows,cols) return i:clone(rows):mid(cols) end

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

function Sym.mid(i) return i.mode end
function Num.mid(i) return i.mu end

function Num.div(i) return i.sd end
function Sym.div(i,  e)
  e=0; for _,n in pairs(i.all) do e=e + n/i.n*math.log(n/i.n,2) end
  return -e 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 = is.missing(a) and is.missing(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     is.missing(a) then b=i:norm(b); a=b<.5 and 1 or 0
  elseif is.missing(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,left,right,lefts,rights,c,mid)
  function project(row,a,b)
    a= dist(i,left,row)
    b= dist(i,right,row)
    return {(a^2 + c^2 - b^2)/(2*c), row}
  end ----------------------
  some  = many(rows,        the.some)
  left  = furthest(i,any(some), some)
  right = furthest(i,left,      some)
  c     = dist(i,left,right)
  lefts,rights = {},{}
  for n, projection in pairs(sort(map(rows,project),firsts)) do
    if n==#rows//2 then mid=row end
    push(n <= #rows//2 and lefts or rights, projection[2]) end
  return lefts, rights, left, right, mid, c  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, here.left, here.right, here.mid = 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,format,t,pre,   front)
  if t then
    pre=pre or ""
    front = fmt("%s%s",pre,#t.all)
    if not t.lefts and not t.rights then
      print(fmt("%-20s%s",front, o(rnds(mids(i,t.all),format))))
    else
      print(front)
      clusters(i,format,t.lefts, "|".. pre)
    clusters(i,format,t.rights,"|".. pre) end end end
---       _| . _  _ _  _  _|_ . _  _
---      (_||_\(_|  (/_   |  |/_(/_

local merge,merged,spans,bestSpan
function Sym.spans(i, j)
  local xys,all,one,last,x,y,n = {}, {}
  for x,n in pairs(i.all) do push(xys, {x,"lefts",n}) end
  for x,n in pairs(j.all) do push(xys, {x,"rights",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=Sym(i.at,i.name)}) 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,"lefts",1}) end
  for _,n in pairs(j.all) do push(xys, {n,"rights",1}) end
  one = {lo=lo, hi=lo, all=Sym(i.at,i.name)}
  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=one.all:clone()}) 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 = a.all:merged(b.all)
      if    both
      then  a = {lo=a.lo, hi=b.hi, all=both}
            j = j + 1 end end
    push(now,a) end
  return #now == #b4 and b4 or merge(now) end

function Sym.merge(i,j,    k)
  k = i:clone()
  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
  return k end

function Sym.merged(i,j,   k,ei,ej,ek)
  k = i:merge(j)
  ei, ej, ek= i:div(), j:div(), k:div()
  if ek*.99 <= (i.n*ei + j.n*ej)/k.n then return k end end

function spans(egs1,egs2,      spans,tmp,col1,col2)
  spans = {}
  for c,col1 in pairs(egs1.x) do
    col2 = egs2.x[c]
    tmp = col1:spans(col2)
    if #tmp> 1 then
      for _,one in pairs(tmp) do push(spans,one) end end end
  return spans end

function bestSpan(spans)
  local divs,ns,n,div,stats,dist2heaven = Num(), Num()
  function dist2heaven(s) return {((1 - n(s))^2 + (0 - div(s))^2)^.5,s} end
  function div(s)         return divs:norm( s.all:div() ) end
  function n(s)           return   ns:norm( s.all.n     ) end
  for _,s in pairs(spans) do
    add(divs, s.all:div())
    add(ns,   s.all.n) end
  return sort(map(spans, dist2heaven), firsts)[1][2]  end
---              _       |  _ . _
---     (/_ >< |_)|(_|| | |
---            |

local xplain,xplains,selects,spanShow
function xplain(i,rows,used,
                 stop,here,left,right,lefts0,rights0,lefts1,rights1)
  used=used or {}
  rows = rows or i.all
  here = {all=rows}
  stop = (#i.all)^the.leaves
  if #rows >= 2*stop then
    lefts0, rights0, here.left, here.right, here.mid, here.c  = half(i, rows)
    if #lefts0 < #rows then
      here.selector = bestSpan(spans(i:clone(lefts0),i:clone(rights0)))
      push(used, {here.selector.all.name, here.selector.lo, here.selector.hi})
      lefts1,rights1 = {},{}
      for _,row in pairs(rows) do
        push(selects(here.selector, row) and lefts1 or rights1, row) end
      if #lefts1  > stop then here.lefts  = xplain(i,lefts1,used) end
      if #rights1 > stop then here.rights = xplain(i,rights1,used) end end end
  return here end

function xplains(i,format,t,pre,how,    sel,front)
  pre, how = pre or "", how or ""
  if t then
    pre=pre or ""
    front = fmt("%s%s%s %s",pre,how, #t.all, t.c and rnd(t.c) or "")
    if t.lefts and t.rights then print(fmt("%-35s",front)) else
      print(fmt("%-35s %s",front, o(rnds(mids(i,t.all),format))))
    end
    sel = t.selector
    xplains(i,format,t.lefts,  "|".. pre, spanShow(sel)..":")
    xplains(i,format,t.rights, "|".. pre, spanShow(sel,true) ..":") end end

function selects(span,row,    lo,hi,at,x)
  lo, hi, at = span.lo, span.hi, span.all.at
  x = row[at]
  if is.missing(x) then return true end
  if lo==hi then return x==lo else return lo <= x and x < hi end end

function spanShow(span, negative,   hi,lo,x,big)
  if not span then return "" end
  lo, hi, x, big  = span.lo, span.hi, span.all.name, math.huge
  if    not negative
  then if lo ==  hi  then return fmt("%s == %s",x,lo)  end
       if hi ==  big then return fmt("%s >= %s",x,lo)  end
       if lo == -big then return fmt("%s < %s",x,hi)  end
       return fmt("%s <= %s < %s",lo,x,hi)
  else if lo ==  hi  then return fmt("%s != %s",x,lo)  end
       if hi ==  big then return fmt("%s < %s",x,lo)  end
       if lo == -big then return fmt("%s >= %s",x,hi)  end
       return fmt("%s < %s and %s >= %s", x,lo,x,hi)  end end
```

```
---      __|_ _ _|_ _
---     _\ |  (_| |  _\

-- function Num:same(i,j, xs,ys,       lt,gt)
--   lt,gt  = 0, 0
--   for _,x in pairs(i.all) do
--     for _,y in pairs(i.all) do
--       if y > x then gt = gt + 1 end
--       if y < x then lt = lt + 1 end end end
--   return math.abs(gt - lt)/(#xs * #ys) <= the.cliffs end
--
-- -- ## Significance
-- -- Non parametric "significance"  test (i.e. is it possible to
-- -- distinguish if an item belongs to one population of
-- -- another).  Two populations are the same if no difference can be
-- -- seen in numerous samples from those populations.
-- -- Warning: very
-- -- slow for large populations. Consider sub-sampling  for large
-- -- lists. Also, test the effect size (and maybe shortcut the
-- -- test) before applying  this test.  From p220 to 223 of the
-- -- Efron text  'introduction to the boostrap'.
-- -- https://bit.ly/3iSJz8B Typically, conf=0.05 and b is 100s to
-- -- 1000s.
-- -- Translate both samples so that they have mean x,
-- -- The re-sample each population separately.
-- function bootstrap(y0,z0,my)
--   local x,y,z,xmu,ymu,zmu,yhat,zhat,tobs,ns, bootstraps, confidence
--   bootstraps = my and my.bootstrap or 512
--   confidence = my and my.conf or .05
--   x, y, z, yhat, zhat = Num.new(), Num.new(), Num.new(), {}, {}
--   for _,y1 in pairs(y0) do x:summarize(y1); y:summarize(y1) end
--   for _,z1 in pairs(z0) do x:summarize(z1); z:summarize(z1) end
--   xmu, ymu, zmu = x.mu, y.mu, z.mu
--   for _,y1 in pairs(y0) do yhat[1+#yhat] = y1 - ymu + xmu end
--   for _,z1 in pairs(z0) do zhat[1+#zhat] = z1 - zmu + xmu end
--   tobs = y:delta(z)
--   n = 0
--   for _= 1,bootstraps do
--     if adds(samples(yhat)):delta(adds(samples(zhat))) > tobs
--     then n = n + 1 end end
--   return n / bootstraps >= conf end
--
-- function scottKnot(nums,the,      all,cohen)
--   local mid = function (z) return z.some:mid()
--   end --------------------------------
--   local function summary(i,j,    out)
--     out = copy( nums[i] )
--     for k = i+1, j do out = out:merge(nums[k]) end
--     return out
--   end ---------------------------
--   local function div(lo,hi,rank,b4,       cut,best,l,l1,r,r1,now)
--     best = 0
--     for j = lo,hi do
--       if j < hi  then
--         l   = summary(lo,  j)
--         r   = summary(j+1, hi)
--         now = (l.n*(mid(l) - mid(b4))^2 + r.n*(mid(r) - mid(b4))^2
--               ) / (l.n + r.n)
--         if now > best then
--           if math.abs(mid(l) - mid(r)) >= cohen then
--             cut, best, l1, r1 = j, now, copy(l), copy(r)
--     end end end end
--     if cut and not l1:same(r1,the) then
--       rank = div(lo,    cut, rank, l1) + 1
--       rank = div(cut+1, hi,  rank, r1)
--     else
--       for i = lo,hi do nums[i].rank = rank end end
--     return rank
--   end ------------------------------------------------------
--   table.sort(nums, function(x,y) return mid(x) < mid(y) end)
--   all   = summary(1,#nums)
--   cohen = all.sd * the.iota
--   div(1, #nums, 1, all)
--   return nums 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,lefts,rights)
  i = file2Egs(the.file)
  lefts,rights = half(i, i.all)
  oo(mids(i, lefts))
  oo(mids(i, rights))
  end

function Demo.cluster(   i)
  i = file2Egs(the.file)
  clusters(i,"%.0f",cluster(i)) end

function Demo.spans(    i,lefts,rights)
  i = file2Egs(the.file)
  lefts, rights = half(i, i.all)
  oo(bestSpan(spans(i:clone(lefts), i:clone(rights)))) end

function Demo.xplain(    i,j,tmp,lefts,rights,used)
  i = file2Egs(the.file)
  used={}
  xplains(i,"%.0f",xplain(i, i.all,used))
  map(sort(used,function(a,b)
        return ((a[1] < b[1]) or
                (a[1]==b[1] and a[2] < b[2]) or
                (a[1]==b[1] and a[2]==b[2] and a[3] < b[3]))end),oo) end


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