```lua
#!/usr/bin/env lua
-- vim : ft=lua et sts=2 sw=2 ts=2 :
local help = [[
sl [OPTIONS]
Sublime's unsupervised bifurcation: let's infer minimal explanations.
(c) 2022, Tim Menzies

OPTIONS:
  -D       stack dump on assert fails
  -d   f   data file                   = etc/data/auto93.csv
  -f   F   far                         = .9
  -k   P   max kept items              = 512
  -p   P   distance coefficient        = 2
  -S   P   set seed                    = 10019
  -t   S   start up action (or 'all') = nothing
  -h       show help

KEY: f=filename F=float P=posint S=string ]]
--                                  _.
--                          _.-----'' `\
--              __..-----'''            `.
--             <            `\.           '\
--             :.             `.            `\
--              `:.            I  `.           `-.
--                `:\ P  a      O  `.            `+.
--                 `:. L  i  a    ns`.   __.===::::;)
--            I  L E   `: n   t   ___.__>'::::::a:f/'
--       i          X   `.  _,===:::=-'-=-""""''
--   m      n  a         '-/:::''
--                         ''
--
-- 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 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.

local b4={}; for k,_ in pairs(_ENV) do b4[k]=k end --used later (to find rogues)
local any,azzert,big,cli,fails,first,fmt,goalp,help,ignorep,klassp
local lessp,map,main,many,max,min,morep,new,nump,o,oo,per,push
local r,rows,second,slots,sort,sum,the,thing,things,unpack
local COLS, EGS, NUM, ROWS, SKIP, SOME, SYM = {},{},{},{},{},{},{}

function cli(want,x)
  for n,got in ipairs(arg) do if got==want then
    x = x==false and true or x==true and "false" or arg[n+1] end end
  if x=="false" then return false else return tonumber(x) or x end end

the = {dump = cli("-D", false),
       data = cli("-d", "etc/data/auto93.csv"),
       help = cli("-h", false),
       far  = cli("-f", .9   ),
       keep = cli("-k", 256  ),
       p    = cli("-p", 2    ),
       seed = cli("-S", 10019),
       todo = cli("-t", "nothing")}
--
-- ___ _   _ _   _ ____ ___ _ ____ _   _ ____
-- |__ |   | |\  | |     |  | |  | |\  | [__
-- |   |___| | \ | |___  |  | |__| | \ | ___]
--
-- strings
fmt = string.format

-- maths
big = math.huge
max = math.max
min = math.min
r   = math.random

-- column headers
function goalp(x)   return morep(x) or lessp(x) or klassp(x) end
function ignorep(x) return x:find":$"     end
function klassp(x)  return x:find"!$"     end
function lessp(x)   return x:find"-$"     end
function morep(x)   return x:find"+$"     end
function nump(x)    return x:find"^[A-Z]" end

-- tables
unpack = table.unpack
function any(t)       return t[r(#t)] end
function first(t)     return t[1] end
function many(t,n, u) u={}; for i=1,n do push(u,any(t)) end; return u end
function per(t,p)     return t[ (#t*(p or .5))//1 ] end
function push(t,x)    table.insert(t,x); return x end
function second(t)    return t[2] end
function sort(t,f)    table.sort(t,f); return t end

-- meta
function map(t,f, u)  u={};for k,v in pairs(t) do push(u,f(v)) end; return u end
function sum(t,f, n)  n=0; for _,v in pairs(t) do n=n+f(v)     end; return n end
function slots(t, u)
  u={}
  for k,v in pairs(t) do k=tostring(k);if k:sub(1,1)~="_" then push(u,k) end end
  return sort(u) end

-- print tables, recursively
function oo(t) print(o(t)) end
function o(t)
  if type(t)~="table" then return tostring(t) end
  local key=function(k) return fmt(":%s %s",k,o(t[k])) end
  local u = #t>0 and map(t,o) or map(slots(t),key)
  return '{'..table.concat(u," ").."}" end

-- strings to things
function rows(file,      x)
  file = io.input(file)
  return function()
    x=io.read(); if x then return things(x) else io.close(file) end end end

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

-- errors
fails=0
function azzert(test, msg)
  print(test and "PASS: "or "FAIL: ",msg or "")
  if not test then
    fails=fails+1
    if the.dump then assert(test,msg) end end end

-- objects
function new(k,t)     k.__index=k; k.__tostring=o; return setmetatable(t,k) end
```

```lua
--  _____  _       _____  _____  _____  _____  _____
-- |      |       |     | [____  [____  |      [____
-- |_____ |_____  |     |  ____]  ____] |_____  ____]
--
-- COLS
function COLS.new(k,row,   i)
  i= new(k,{all={},x={},y={},names=row})
  for at,txt in ipairs(row) do  push(i.all, i:col(at,txt)) end
  return i end

function COLS.add(i,t)
  for _,col in pairs(i.all) do col:add( t[col.at] ) end
  return t end

function COLS.col(i,at,txt,     col)
  if ignorep(txt) then return SKIP:new(at,txt) end
  col = (nump(txt) and NUM or SYM):new(at,txt)
  push(goalp(txt) and i.y or i.x, col)
  if klassp(txt) then i.klass = col end
  return col end

-- NUM
function NUM.new(k,n,s)
  return new(k,{n=0,at=n or 0,txt=s or"",has=SOME:new(),ok=false,
                w=lessp(s or "") and -1 or 1, lo=big, hi=-big}) end

function NUM.add(i,x)
  if x ~= "?" then
    i.n = i.n + 1
    if i.has:add(x) then i.ok=false end
    i.lo,i.hi = min(x,i.lo), max(x,i.hi); end end

function NUM.dist(i,x,y)
  if x=="?" and y=="?" then return 1
  elseif x=="?" then y=i:norm(y); x=y<0.5 and 1 or 0
  elseif y=="?" then x=i:norm(x); y=x<0.5 and 1 or 0
  else   x,y = i:norm(x), i:norm(y) end

function NUM.norm(i,x)
  return math.abs(i.hi-i.lo)<1E-9 and 0 or (x-i.lo)/(i.hi - i.lo) end

function NUM.sorted(i)
  if i.ok==false then table.sort(i.has._all); i.ok=true end
  return i.has._all end
return math.abs(x-y) end

-- ROWS
function ROWS.new(k,inits,     i)
  i = new(k,{rows=SOME:new(), cols=nil})
  if type(inits)=="string" then for row in rows(inits) do i:add(row) end end
  if type(inits)=="table"  then for row in inits       do i:add(row) end end
  return i end

function ROWS.add(i,row)
  if   i.cols then i.rows:add( i.cols:add(row) )
  else i.cols = COLS:new(row) end end

function ROWS.clone(i) j= ROWS:new(); j.add(i.cols.names);return j end

function ROWS.dist(i,row1,row2,   d)
  function fun(col) return col:dist(row1[col.at], row2[col.at])^the.p end
  return (sum(i.cols.x, fun)/ #i.cols.x)^(1/the.p) end

function ROWS.far(i,row1,rows)
  function fun(row1) return {i.dist(row1,row2), row2} end
  return unpack(per(sort(map(rows,fun),first), the.far)) end

function ROWS.half(i, top)
  local top,c,x,y,tmp,mid,lefts,rights
  top = top or i
  _,x = top:far(any(i.rows), i.rows)
  c,y = top:far(x,            i.rows)
  tmp = sort(map(i.rows, function(r) return top:project(r,x,y,c) end), first)
  mid = (#i.rows)//2
  lefts, rights = i:clone(), i:clone()
  for at,row in pairs(tmp) do (at <=mid and lefts or rights):add(row[2]) end
  return lefts,rights,x,y,c, tmp[mid] end

function ROWS.project(i, r,x,y,c,     a,b)
  a,b = i.dist(r,x), i.dist(r,y); return {(a^2 + c^2 - b^2)/(2*c), r} end

-- SKIP
function SKIP.new(k,n,s) return new(k,{n=0,at=at or 0,txt=s or""}) end
function SKIP.add(i,x)   return x end

-- SOME
function SOME.new(k,keep) return new(k,{n=0,_all={}, keep=keep or the.keep}) end
function SOME.add(i,x)
  i.n = i.n+1
  if     #i._all < i.keep then push(i._all,x)          ; return i._all
  elseif r()     < i.keep/i.n then i._all[r(#i._all)]=x; return i._all end end

-- SYM
function SYM.new(k,n,s)  return new(k,{n=0,at=n or 0,txt=s or"",has={}}) end
function SYM.dist(i,x,y) return(x=="?" and y=="?" and 1) or(x==y and 0 or 1) end
function SYM.add(i,x,inc)
  if x ~= "?" then
    inc = inc or 1
    i.n = i.n + inc
    i.has[x] = inc + (i.has[x] or 0) end end
```

```lua
--  _____  _____  _   _  _____  _____
-- |     \ |      |\/|  |     | [____
-- |_____/ |_____ |    | |_____|  ____]
--
function EGS.nothing() return true end
function EGS.the()     oo(the) end
function EGS.rand()    print(r()) end
function EGS.clone( r,s)
  r = ROWS:new(the.data)
  s = r:clone()
  end

function EGS.data( r)
  r = ROWS:new(the.data)
  azzert(r.cols.x[1].hi == 8, "data.columns") end

function EGS.dist( r,rows,n)
  r = ROWS:new(the.data)
  rows = r.rows._all
  n = NUM:new()
  for _,row in pairs(rows) do n:add(r:dist(row, rows[1])) end
  oo(r.cols.x[2]:sorted()) end

-- start-up
if the.help then print(help) else
  local b4={}; for k,v in pairs(the) do b4[k]=v end
  for _,todo in pairs(the.todo=="all" and slots(EGS) or {the.todo}) do
    for k,v in pairs(b4) do the[k]=v end
    math.randomseed(the.seed)
    if type(EGS[todo])=="function" then EGS[todo]() end end end

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