```lua
#!/usr/bin/env lua
-- vim : filetype=lua ts=2 sw=2 et :
--             ________
--            /\_____  \
--      __    \/____//'/'
--    /'__`\       //'/'
--   /\  __/      //'/'___
--   \ \____\     /\_______\
--    \/____/     \/_______/
--
local b4={}; for k,_ in pairs(_ENV) do b4[k]=k end
local cli, coerce
local function defaults() return cli{
        debug=false,
        file="../../data/auto93.csv",
        p=2,
        seed=10019} end

-- command line management
function coerce(x)
  if x=="true" then return true elseif x=="false" then return false end
  return tonumber(x) or x end

function cli(t)
  for flag,x in pairs(t) do
    for n,word in ipairs(arg) do
      if flag:match("^"..word:sub(2).."*") then
        x= coerce(x==true and"false" or x==false and"true" or arg[n+1]) end end
    t[flag]=x end
  return t end

local THE=defaults()
--            __
--     |   | |__)
--     |__ | |__)
--
-- random stuff
local function rand(lo,hi,     mult,mod)
  lo, hi = lo or 0, hi or 1
  THE.seed = (16807 * THE.seed) % 2147483647
  return lo + (hi-lo) * THE.seed / 2147483647 end
local function randi(lo,hi) return math.floor(0.5 + rand(lo,hi)) end

-- string stuff
local fmt = string.format
local function red(s)    return fmt("\27[1m\27[%sm%s\27[0m",31,s) end
local function green(s)  return fmt("\27[1m\27[%sm%s\27[0m",32,s) end
local function yellow(s) return fmt("\27[1m\27[%sm%s\27[0m",33,s) end
-- list stuff
local function push(t,x) table.insert(t,x); return x end
local function sort(t,f) table.sort(t,f);   return t end
local function map(t,f,  u)
  u={};for k,v in pairs(t) do u[1+#u] =f(v)end; return u end
local function keys(t,   u)
  u={};for k,_ in pairs(t) do u[1+#u]=k end; return sort(u) end

-- printing stuff
local function o(t,    u)
  if   type(t) ~= "table" then return tostring(t) end
  function one(k) return fmt(":%s %s",yellow(k), o(t[k])) end
  u = #t>0 and map(t,o) or map(keys(t),one)
  return fmt("%s{%s}",green(t._is or ""),table.concat(u,",")) end
local function out(x) print(o(x)) end

-- interacting with operating system
local function csv(file,   x,coerce,line)
  function line(x,  t)
    t={}; for y in x:gsub("[\t ]*",""):gmatch("([^,]+)" do push(t,coerce(y)) end
    return t end
  file = io.input(file)
  return function(   x)
    x = io.read()
    if x then return line(x) else io.close(file) end end end

-- oo stuff
local function as(mt,x) return setmetatable(x,mt) end
local function is(s, obj)
  obj = {_is=s, __tostring=obj}; obj.__index = obj
  return as({__call=function(_,...) return obj.new(...) end},obj) end

--      __        __
--     (_  |_/ | |__)
--     __) | \ | |
--
local SKIP=is"SKIP"
function SKIP.new(inits,txt,at)
  return as(SKIP, {at=at or 0, txt=txt or ""}) end
function SKIP.add(i,x) return x end
--      __
--     (_  \_/ |\/|
--     __)  |  |  |
--
local SYM=is"SYM"
function SYM.new(inits,txt,at, i)
  i= as(SYM, {at=at or 0, txt=txt or "", has={}, most=0,mode=nil})
  for _,x in pairs(inits or {}) do i:add(x) end
  return i end

function SYM.add(i,x)
  if x~="?" then
      i.has[x] = 1 + (i.has[x] or 0)
      if i.has[x] > i.most then i.mode, i.most= x, i.has[x] end end
  return end

function SYM.mid(i) return i.mode end
function SYM.spread(i,   e,n)
  n=0; for _,v in pairs(i.has) do n = n+v end
  e=0; for _,v in pairs(i.has) do e = e- v/n * math.log(v/n,2) end
  return e end
--
--     |\ | /  \ |\/|
--     | \| \__/ |  |
--
local NUM=is"NUM"
function NUM.new(inits,txt,at, i)
  i= as(NUM, {at=at or 0, txt=txt or "", w= (txt or ""):find"-" and -1 or 1,
                has={},      ready=false,   lo=math.huge, hi=-math.huge})
  for _,x in pairs(inits or {}) do i:add(x) end
  return i end

function NUM.add(i,x)
  if i ~= "?" then
    push(i.has,x)
    i.ready=false
    if x> i.hi then i.hi=x elseif x<i.lo then i.lo=x end
    return x end end

function NUM.all(i)
  if not i.ready then table.sort(i.has); i.ready=true; end
  return i.has end
function NUM.mid(i) return i:per(.5) end
function NUM.norm(i,x)
  return x=="?" and x or math.abs(i.hi - i.lo) < 1E-9 and 0 or (x-i.lo)/(i.hi-i.lo) end
function NUM.per(i,p,  t) t = i:all(); p=p*#t//1; return t[p<1 and 1 or p] end
function NUM.spread(i) return math.abs(i:per(.9) - i:per(.1))/ 2.56 end
```

```lua
local EG=is"EG"
function EG.new(cells) return as(EG,{cells=cells}) end
function EG.cols(i,all)
  return map(all,function(_,c) return i.cells[c.at] end) end

function EG.dist(i,other,cols,   a,b,d,n,inc)
  d,n = 0,0
  for _,col in pairs(cols) do
    a,b = i.cells[col.at], j.cells[col.at]
    inc = a=="?" and b=="?" and 1 or col:dist(a,b)
    d   = d + inc^THE.p
    n   = n + 1 end
  return (d/n)^(1/THE.p) end

-- Sorting
function EG.better(i,j,cols,     e,n,a,b,s1,s2)
  n,s1,s2,e = #cols, 0, 0, 2.71828
  for _,col in pairs(cols) do
    a  = col:norm(i.cells[num.at])
    b  = col:norm(j.cells[num.at])
    s1 = s1 - e^(col.w * (a-b)/n)
    s2 = s2 - e^(col.w * (b-a)/n) end
  return s1/n < s2/n end

--        __             __        __
--       (_   /\  |\/|  |__)  |   |_
--       __) /--\ |  |  |     |__ |__
--
local SAMPLE=is"SAMPLE"
function SAMPLE.new(eg)
  return as(SAMPLE,{egs={}, cols={}, xs={}, ys={},names=nil}) end
function SAMPLE.clone(i,inits,    j)
  j= SAMPLE(i.names)
  for _,eg in pairs(inits or {}) do j:add(eg) end; return j end

function SAMPLE.add(i, eg)
  eg = eg.cells and eg.cells or eg
  if i.names then
    for k,v in pairs(i.nums) do num1(v, eg[v]) end
    for k,v in pairs(i.syms) do sym1(v, eg[v]) end
    push(i.egs,{cells=eg}) end
  function headers(eg)
    for k,v in pairs(eg) do
      if not v:find":" then
        if v:find"+" or v:find"-" then i.ys[k]=v else i.xs[k]=v end
        if v:find"^[A-Z]" then
           local w = v:find"-" and 1 or -1
           i.nums[k]={has={},ok=false,w=w, lo=math.huge, hi=-math.huge}
        else
           i.syms[k]={has={},mode=0,most=0} end end end
     return eg end
  if i.names then data(eg) else i.names = headers(eg) end
  return i end

local group={}
function group.dist(i,eg1,eg2)
  local d,n,inc,sym1,num1 = 0,1E-9
  function sym1(a,b) return a=="?" and b=="?" and 1 or a==b and 0 or 1 end
  function num1(a,b)
    if     a=="?" and b=="?" then return 1
    elseif a=="?"            then a = b>.5 and 0 or 1
    elseif b=="?"            then b = a>.5 and 0 or 1 end
    return math.abs(a-b) end
  for k,_ in pairs(i.xs) do
    if i.num[k] then
      inc=  num1(norm(num,eg1.cells[k]), norm(num,eg2.cells[k]))
    else
      inc=  sym1(eg.cells[k], eg2.cells[k]) end
    d = d+ inc^THE.p
    n = n + 1 end
  return (d/n)^(1/THE.p) end

function group.neighbors(eg1,egs,cols,        dist_eg2)
  dist_eg2 = function(_,eg2) return {eg1:dist(eg2,cols or self.cols.xs),eg2} end
  return sort(map(egs or self.egs,dist_eg2),ones) end

function SAMPLE:distance_farEg(eg1,egs,cols,    tmp)
  tmp = self:neighbors(eg1, egs, cols)
  tmp = tmp[#tmp*THE.Far//1]
  return tmp[2], tmp[1] end

--       __              __        __  __
--      |_  \_/  /\  |\/| |__) |   |_  (_
--      |__ / \ /--\ |  | |    |__ |__ __)
--
local go,fails = {}, 0
local function run(x)
  map(x and {x} or keys(go), function(k)
      THE = defaults();
      ok,msg =pcall(go[k])
      if ok then print(fmt(green("-- PASS [%s]"),k))
            else print(fmt(red(  "-- FAIL [%s]:%s"),k,msg:gsub("^.*:",""))); fails=fails+1 end en
d) end

function go.the() assert(THE.p==3,"p is two?") end

run()
os.exit(fails)
```