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

---- ---- ---- ---- Config
local     the
m.config={the}
the= {
      bins   = 7,
      far    = .95,
      files  = "../../data/auto93.csv",
      go     = "nothing",
      min    = .5,
      nums   = 512,
      p      = 2,
      seed   = 10019
      }
---- ---- ---- ---- Library
m.lib={}
m.lib.maths={}
m.lib.lists={}
m.lib.read={}
m.lib.write={}
---- ---- ---- Maths
local         rand,rnd
m.lib.maths={rand,rnd}

rand=math.random

function rnd(x, places)
  local mult = 10^(places or 2)
  return math.floor(x * mult + 0.5) / mult end

---- ---- ---- Lists
local         rev,sort,lt,gt,push,map,any,per
m.lib.lists={rev,sort,lt,gt,push,map,any,per}

function rev(t)
  for i=1, math.floor(#t / 2) do t[i],t[#t-i+1] = t[#t-i+1],t[i] end
  return t end

function sort(t,f) table.sort(t,f); return t end
function lt(x) return function(a,b) return a[x] < b[x] end end
function gt(x) return function(a,b) return a[x] > b[x] end end

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

function map(t,f)
  local u={}; for _,v in pairs(t) do u[1+#u]=f(v) end; return u end

function any(a) return a[rand(#a)] end

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

---- ---- ---- Print
local         fmt,chat,cat
m.lib.print={fmt,chat,cat}

fmt = string.format

function chat(t) print(cat(t)) return t end
function cat(t,    show,u)
  if type(t)~="table" then return tostring(t) end
  function show(k,v) return #t==0 and fmt(":%s %s",k,v) or tostring(v) end
  u={}; for k,v in pairs(t) do u[1+#u]=show(k,v) end
  return (t._is or "").."{"..table.concat(#t==0 and sort(u) or u," ").."}" end

---- ---- ---- Read
local         coerce,cli,words,lines,csv,csv2data
m.lib.read={coerce,words,lines,csv,csv2data}

-- Try reading `str` as a boolean, then int, then float, then string.
function coerce(str)
  str = str:match"^%s*(.-)%s*$"
  if str=="true" then return true elseif str=="false" then return false
  else return math.tointeger(str) or tonumber(str) or str end  end

-- Read update for `slot` of table from command line flag `-s` or `--slot`.
-- If slot's is a boolean, this code flips old value.
function 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] =  coerce(v) end
  return t end

-- Split  `str` on `sepstr`, filter each part through `fun`, return the resulting list
.
function words(str,sepstr,fun,      t)
  fun = fun or function(z) return z end
  sep = fmt("([^%s]+)",sepstr)
  t={};for x in str:gmatch(sep) do t[1+#t]=fun(x) end;return t end

-- Read lines from `filestr`, closing stream at end. Call `fun` on each line.
function lines(filestr, fun)
  local src = io.input(filestr)
  while true do
    local str = io.read()
    if not str then return io.close(src) else fun(str) end end end

-- Read lines from `filestr`, converting each into words, passing that to `fun`.
function csv(filestr, fun)
  lines(filestr,
    function(t) fun(words(t,",",coerce)) end) end

-- Read `filestr` into a DATA object. Return that object.
function csv2data(filestr,data)
  data=DATA()
  csv(filestr, function(t) row(data,t) end)
  return data end

---- ---- ---- ---- Types
local    is,COL,ROW,ABOUT,DATA
m.types={is,COL,ROW,ABOUT,DATA}

is={num    = "^[A-Z]",  -- numeric cols start with uppercase
    goal   = "[!+-]$",  -- !=klass, [+,-]=maximize,minimize
    klass  = "!$",      -- klass if "!"
    skip   = ":$",      -- skip if ":"
    less   = "-$"}      -- minimize if "-"

function COL(num,txt)
  txt=txt or ""
  return {n    = 0,
          at   = num or 0,
          txt  = txt,
          nump = txt:find(is.num),
          w    = txt:find(is.less) and -1 or 1,
          ok   = true,
          has = {}} end

function ROW(about, t)
  return {about=about,cells=t,cooked=t} end

function ABOUT(strs)
  local about = {names=strs,all={}, x={}, y={}, klass=nil}
  for at,txt in pairs(strs) do
    local one = push(about.all, COL(at,txt))
    if not txt:find(is.skip) then
      push(txt:find(is.goal) and about.y or about.x, one)
      if txt:find(is.klass) then about.klass=one end end end
  return about end

function DATA() return {has={},about=nil} end

---- ---- ---- ---- Methods
---- ---- ---- Update
local     add,adds,row,clone
m.update={add,adds,row,clone}

function add(col,x)
  if x ~= "?" then
    col.n = col.n + 1
    if   not col.nump
    then col.has[x] = 1 + (col.has[x] or 0)
    else local pos
         if     #col.has < the.nums      then pos= (#col.has) + 1
         elseif rand() < the.nums/self.n then pos= rand(#col.has) end
         if pos then
           col.ok=false  -- the `kept` list is no longer in sorted order
           col.has[pos]=x end end end end

function adds(about,t)
  t = t.cells and t or ROW(about,t)
  for _,cols in pairs{about.x,about.y} do
    for _,col in pairs(cols) do add(col, t.cells[col.at]) end end
  return t end

function row(data,t)
  if   data.about
  then push(rows.has, adds(data.about,t))
  else data.about = ABOUT(t) end end

function clone(data0,t)
  local data = DATA()
  row(data1, data0.about.names)
  for _,row1 in pairs(t or {}) do row(data1,row1) end
  return data1 end

---- ---- ---- Query
local     has,norm,mid,div,stats,better
m.query={has,norm,mid,div,stats,better}

function has(col)
  if col.nump and not col.ok then table.sort(col.has); col.ok=true end
  return col.has end

function norm(col,x)
  local a= has(col)
  return a[#a] - a[1] < 1E-9 and 0 or (x-a[1])/(a[#a]-a[1]) end

function mid(col)
  if col.nump then return per(has(col),.5)  end
  local mode,most= -1,nil
  for x,n in pairs(col.has) do if n>most then mode,most=x,n end end
  return mode end

function div(col)
  if col.nump then return (per(has(col),.9)-per(has(col),.1))/2.58 end
  local ent=0
  for _,n in pairs(col.has) do
    if n>0 then ent=ent-n/col.n*math.log(n/col.n,2) end end
  return ent end

function stats(data,places,f,about,   u)
  f =  f or mid
  about = about or data.about.y
  u={}; for k,col in pairs(about.y) do
    u.n=col.n; u[col.txt]=rnd(f(col),places) end;
  return u end

function better(row1,row2)
  local s1,s2,d,n=0,0,0,0
  local ys,e = row1.about.y,math.exp(1)
  for _,col in pairs(ys) do
    x,y= row1.cells[col.at], row2.cells[col.at]
    x,y= norm(col,x), norm(col,y)
    s1 = s1 - e^(col.w * (x-y)/#ys)
    s2 = s2 - e^(col.w * (y-x)/#ys) end
  return s1/#ys < s2/#ys end

---- ---- ---- Dist
local    dist,around,far,half,halfsort
m.dist={dist,around,far,half,halfsort}

function dist(row1,row2)
  local d,n,x,y,dist1=0,0
  function dist1(col,x,y)
    if x=="?" and y=="?" then return 1 end
    if   col.nump
    then if     x=="?" then y=norm(col,y); x=y<.5 and 1 or 0
         elseif y=="?" then x=norm(col,x); y=x<.5 and 1 or 0
         else   x,y = norm(col,x), norm(col,y) end
         return math.abs(x-y)
    else return (x=="?" or y=="?") and 1 or x==y and 0 or 1 end
  end ----------------

  for _,col in pairs(row1.about.x) do
    x,y = row1.cells[col.at], row2.cells[col.at]
    d = d+dist1(col,x,y)^the.p
    n = n + 1 end
  return (d/n)^(1/the.p) end

function around(row1,rows)
  return sort(map(rows, function(row2) return {row=row2,d=dist(row1,row2)} end),
              lt"d") end

function half(rows, rowAbove)
  local As,Bs,A,B,c,far,project = {},{}
  function far(row)     return per(around(row,rows), the.far).row end
  function project(row) return {row=row,
                                x=(dist(row,A)^2 - c^2 + dist(row,B)^2)/(2*c)} end
  A= rowAbove or far(any(rows))
  B= far(A)
  c= dist(A,B)
  for n,rd in pairs(sort(map(rows, project),lt"x")) do
    push(n < #rows/2 and As or Bs, rd.row) end
  return A,B,As,Bs,c end

function halfsort(rows,rowAbove,stop,t)
  stop = stop or (#rows)^the.min
  out  = out or {}
  if   #rows < stop
  then for _,row in pairs(rows) do push(out,row) end
       return out
  else local A,B,As,Bs = half(rows,rowAbove)
       if better(A,B) then
         for _,row in pairs(reverse(Bs)) do push(out,row) end
         return halfsort(reverse(As),A,stop,out)
       else
         for _,row in pairs(As) do push(out,row) end
         return halfsort(Bs,B,stop,out) end end  end

---- ---- ---- ---- Start
local go={}
function go.the() chat(the); return true end

function go.one(data1,rows2)
  rows1=csv2data("../../data/auto93.csv")
  print("mid1", cat(stats(rows1,2,mid)))
  print("div1", cat(stats(rows1,2,div)))
  rows2=clone(rows1,rows1.has)
  print("mid2", cat(stats(rows2,2,mid)))
  print("div2", cat(stats(rows2,2,div)))
  return true
  end

function go.dist(rows,row1,row2)
  rows= csv2data("../../data/auto93.csv")
  for i = 1,20 do
    row1=any(rows.has)
    row2=any(rows.has)
    print(dist(row1,row2)) end
  return true end

local fails=0
local function run(str)
  if type(go[str])=="function" then
    local saved={};for k,v in pairs(the) do saved[k]=v end
    math.randomseed(the.seed)
    if true ~= go[str]() then fails=fails+1; print("FAIL",str) end
    for k,v in pairs(saved) do the[k]=v end  end end

if pcall(debug.getlocal,4,1) then
  return {config=config,types=types,update=update,query=query,dist=dist,lib=lib}
end
the=cli(the)
for k,_ in pairs(the.go=="all" and go or {[the.go]=the.go}) do run(k) end
for k,v in pairs(_ENV) do if not b4[k] then print("?",k,type(v)) end end
os.exit(fails)
```