```lua
-- In this code:
-- - Each line is usually 80 chars (or less)
-- - Two spaces before function argumnets denote optionals.
-- - Four spaces before function argumnets denote local variables..
-- - Private functions start with `_`
-- - Arguments of private functions do anything at all
-- - Local variables inside functions do anything at all
-- - Arguments of public functions use type hints
--   - Variable  `x` is is anything
--   - Prefix `is` is a boolean
--   - Prefix `fun` is a function
--   - Prefix `f` is a filename
--   - Prefix `n` is a string
--   - Prefix `s` is a string
--   - Prefix `c` is a column index
--   - `col` denotes `num` or `sym`
--   - `x` is anything (table or number of boolean or string
--   - `v` is a simple value (number or boolean  or  string)
--   - Suffix `s` is a list of things
--   - Tables are `t` or, using the above, a table of numbers would be `ns`
--   - Type names are lower case versions of constuctors. so in this code,
--     `Cols`, `data`, `num`,`sym` are made by functions `Cols` `Data`, `Num`, `Sym`
local l=require"lib0"
local the=l.settings([[
SAM0 : semi-supervised multi-objective explainations
(c) 2022 Tim Menzies <timm@ieee.org> BSD-2 license

USAGE: lua eg0.lua [OPTIONS]

OPTIONS:
 -e  --eg     start-up example         = nothing
 -h  --help   show help                = false
 -n  --nums   how many numbers to keep = 256
 -p  --p      distance coeffecient     = 2
 -s  --seed   random number seed       = 10019]])

local copy,csv,o,oo = l.coerce,l.copy,l.csv,l.o,l.oo
local per,push      = l.per, l.push

local adds,add, dist,div,mid,nums,read, record
local Cols, Data, Num, Row, Sym

---- ---- ---- ---- Classes
-- Holder of `rows` and their sumamries (in `cols`).
function Data() return {cols=nil,  rows={}} end

-- Hoder of summaries
function Cols() return {klass=nil, names={}, nums={}, x={}, y={}, all={}} end

-- Summary of a stream of symbols.
function Sym(c,s)
  return {n=0,at=c or 0, name=s or "", _has={}} end

-- Summary of a stream of numbers.
function Num(c,s)
  return {n=0,at=c or 0, name=s or "", _has={},
          isNum=true, lo= math.huge, hi= -math.huge, sorted=true,
          w=(s or ""):find"-$" and -1 or 1} end

-- Hold one record, in `cells` (and `cooked` is for discretized data).
function Row(t) return {cells=t, cooked=copy(t)} end

---- ---- ---- ---- Data Functions
---- ---- ---- Update
-- Add one or more items, to `col`. From Num, keep at most `nums` items.
function adds(col,t) for _,v in pairs(t) do add(col,v) end; return col end
function add(col,v)
  if v~="?" then
    col.n = col.n + 1
    if not col.isNum then col._has[v] = 1 + (col._has[v] or 0) else
       col.lo = math.min(v, col.lo)
       col.hi = math.max(v, col.hi)
       local pos
       if     #col._has < the.nums           then pos = 1 + (#col._has)
       elseif math.random() < the.nums/col.n then pos = math.random(#col._has) end
       if pos then col.sorted = false
                   col._has[pos] = tonumber(v) end end end end

---- ---- ---- Query
-- Return kept numbers, sorted.
function nums(num)
  if not num.sorted then table.sort(num._has); num.sorted=true end
  return num._has end

-- Diversity (standard deviation for Nums, entropy for Syms)
function div(col)
  if  col.isNum then local a=nums(col); return (per(a,.9)-per(a,.1))/2.58 else
    local function fun(p) return p*math.log(p,2) end
    local e=0
    for _,n in pairs(_has) do if n>0 then e=e-fun(n/col.n) end end
    return e end end

-- Central tendancy (median for Nums, mode for Syms)
function mid(col)
  if col.isNum then return per(nums(col),.5) else
    local most,mode = -1
    for k,v in pairs(_has) do if v>most then most,mode=k,v end end
    return mode end end

---- ---- ---- ---- Data functions
---- ---- ---- Create
-- Processes table of name strings (from row1 of csv file)
local function _head(sNames)
  local cols = Cols()
  cols.names = namess
  for c,s in pairs(sNames) do
    local col = push(cols.all, -- Numerics start with Uppercase.
                     (s:find"^[A-Z]*" and Num or Sym)(c,s))
    if not s:find":$" then -- some columns are skipped
      push(s:find"[!+-]" and cols.y or cols.x, col) -- some cols are goal cols
      if s:find"!$"    then cols.klass=col end end end
  return cols end

-- if `src` is a string, read rows from file; else read rows from a `src`  table
function read(src)
  local data,fun=Data()
  function fun(t) if data.cols then record(data,t) else data.cols=_head(t) end end
  if type(src)=="string" then csv(src,fun)
                         else for _,t in pairs(src or {}) do fun(t) end end
  return data end

---- ---- ---- Update
-- Add a new `row` to `data`, updating the `cols` with the new values.
function record(data,xs)
  local row= push(data.rows, xs.cells and xs or Row(xs)) -- ensure xs is a Row
  for _,todo in pairs{data.cols.x, data.cols.y} do
    for _,col in pairs(todo) do
      add(col, row.cells[col.at]) end end end

---- ---- ---- Query
-- For `showCols` (default=`data.cols.x`) in `data`, report `fun` (default=`mid`).
function stats(data,  showCols,fun,     t)
  showCols, fun = showCols or data.cols.y, fun or mid
  t={}; for _,col in pairs(showCols) do t[col.name]=fun(col) end; return t end

---- ---- ---- ---- Distance functions
-- Distance between two values`v1,v2`  within `col`
local function _dist1(col,  v1,v2)
  if   v1=="?" and v2=="?" then return 1 end
  if  not col.isNum        then return v1==v2 and 0 or 1 end
  local function norm(n) return (n-col.lo)/(col.hi-col.lo + 1E-32) end
  if     v1=="?" then v2=norm(v2); v1 = v2<.5 and 1 or 0
  elseif v2=="?" then v1=norm(v1); v2 = v1<.5 and 1 or 0
  else   v1,v2 = norm(v1), norm(v2) end
  return  maths.abs(v1-v2) end

-- Distance between two rows (returns 0..1)
function dist(data,t1,t2)
  local d = 0
  for _,col in pairs(data.cols.x) do
    d = d + _dist1(col, t1.cells[col.at], t2.cells[col.at])^the.p end
  return (d/#data.cols.x)^(1/the.p) end

return {the=the,add=add,adds=adds,mid=mid,div=div,dist=dist,
        nums=nums,record=record,
        Cols=Cols,Num=Num, Sym=Sym, Data=Data}
```