```lua
local b4={}; for k,_ in pairs(_ENV) do b4[k]=k end
local coerce,csv,o,oo,push
local cols,dist,data,header,norm,row

-- - Each line is usually 80 chars (or less)
-- - 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`

---- ---- ---- ---- Data
---- ---- ---- 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

---- ---- ---- Data functions
-- Add one or more items, to `col`.
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
        push(col._has,v)
        col.sorted = false
        col.hi = math.max(col.hi, v)
        col.lo = math.min(col.lo, v)
        if col.n % 2*256 == 0 then sorted(col) end
        end end end

function sorted(num)
  if not num.sorted then
    table.sort(num._has)
    if #num._has > 256+1 then
      local tmp={};for i=1,#num._has,#num._has//256 do push(tmp,num._has[i]) end
      num._has= tmp end end
  num.sorted = true
  return num. _has end

-- Add a new `row` to `data`.
function row(data,t)
  push(data.rows,t)
  for _,todo in pairs{data.cols.x, data.cols.y} do
    for _,col in pairs(todo) do
      add(col, t[col.at]) end end end

-- Processes table of name strings (from row1 of csv file)
local function _header(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 load(src)
  local data,fun=Data()
  function fun(t) if data.cols then row(data,t) else data.cols=_header(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

-- function stats(data,cols)
--   for at,col in pairs(cols or data.cols.y) do
--     for _,row in pairs(data.rows) do
--       if
--
---- ---- ---- ---- Cluster
-- Distance between two rows (returns 0..1)
function dist(data,t1,t2)
  local d = 0
  for _,col in pairs(data.cols.x) do
    if   v1=="?" and v2=="?"
    then d = d + 1
    else local v1 = norm(col,t1[col.at])
         local v2 = norm(col,t2[col.at])
         if   not col.isNum
         then d = d + (v1==v2 and 0 or 1)
         else if v1=="?" then v1 = v2<.5 and 1 or 0 end
              if v2=="?" then v2 = v1<.5 and 1 or 0 end
              d = d + maths.abs(v1-v2)^2 end end end
  return (d/data.cols.nx)^.5 end

-- Numbers get normalized 0..1. Everything esle normalizes to itself.
function norm(col,v)
  if v=="?" or not col.isNum then return v else
    local lo = col.lo[c]
    local hi = col.hi[c]
    return (hi - lo) <1E-9 and 0 or (v-lo)/(hi-lo) end end
---- ---- ---- ---- Lib
---- ---- ---- Lists
-- Add `x` to a list. Return `x`.
function push(t,x) t[1+#t]=x; return x end

---- ---- ---- Strings
-- `oo` prints the string from `o`.
-- `o` generates a string from a nested table.
function oo(t) print(o(t)) return t end
function o(t)
  if type(t) ~=  "table" then return tostring(t) end
  local function show(k,v)
    if not tostring(k):find"^[A-Z]"  then
      v=o(v)
      return #t==0 and string.format(":%s %s",k,v) or tostring(v) end end
  local u={}; for k,v in pairs(t) do u[1+#u] = show(k,v) end
  table.sort(u)
  return (t._is or "").."{"..table.concat(u," ").."}" end

-- Convert string to something else.
function coerce(s)
  local function coerce1(s)
    if str=="true"  then return true end
    if str=="false" then return false end
    return s end
  return tonumber(s) or coerce1(s:match"^%s*(.-)%s*$") end

-- Iterator over csv files. Call `fun` for each record in `fname`.
function csv(fname,fun)
  local src = io.input(fname)
  while true do
    local s = io.read()
    if not s then return io.close(src) else
      local t={}
      for s1 in s:gmatch("([^,]+)") do t[1+#t]=coerce(s1) end
      fun(t) end end end
--- Cluster -----------------------------------------------------
local eg={}
function eg.load() oo(load("../../data/auto93.csv").cols); return true end
function eg.dist() oo(load("../../data/auto93.csv").cols); return true end

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