```lua
local help=[[
SEEN : summarized csv file
(c) 2022 Tim Menzies <timm@ieee.org> BSD-2 license

USAGE: lua seen.lua [OPTIONS]

OPTIONS:
 -e  --eg    start-up example     = nothing
 -f  --file  file with csv data   = ../data/auto93.csv
 -h  --help  show help            = false
 -n  --nums  number of nums to keep = 512
 -s  --seed  random number seed   = 10019]]

-- ## Misc routines
-- ### Linting code
-- Find rogue locals.
local b4={}; for k,v in pairs(_ENV) do b4[k]=v end
local function rogues()
  for k,v in pairs(_ENV) do if not b4[k] then print("?",k,type(v)) end end end

-- ### Handle Settings
-- Parse `the` config settings from `help`.
local the={}
local function coerce(s)
  local function coerce1(s1)
    if s1=="true"  then return true end
    if s1=="false" then return false end
    return s1 end
  return math.tointeger(s) or tonumber(s) or coerce1(s:match"^%s*(-)%s*$") end

help:gsub("\n[-][%S]+[%s]+[-]([%S]+)[^\n]+= ([%S]+)",
          function(k,x) the[k]=coerce(x) end)

-- Update settings from values on command-line flags. Booleans need no values
-- (we just flip the defeaults).
local 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
  if t.help then os.exit(print("\n"..help.."\n")) end
  return t end

-- ### Strings
-- `o` generates a string from a nested table.
local function o(t)
  if type(t) ~= "table" then return tostring(t) end
  local function show(k,v)
    if not tostring(k):find"^_"  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
  if #t==0 then table.sort(u) end
  return (t._is or "").."{"..table.concat(u,"").."}" end

-- `oo` prints the string from `o`.
local function oo(t) print(o(t)) return t end

-- ### Lists
-- Deepcopy
local function copy(t)
  if type(t) ~= "table" then return t end
  local u={}; for k,v in pairs(t) do u[k] = copy(v) end
  return setmetatable(u,getmetatable(t))  end

-- Return the `p`-th thing from the sorted list `t`.
local function per(t,p)
  p=math.floor(((p or .5)*#t)+.5); return t[math.max(1,math.min(#t,p))] end

-- Add to `t`, return `x`.
local function push(t,x) t[1+#t]=x; return x end

-- ## Call `fun` on each row.
local 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
```

```lua
-- -------------------------------------
-- ## Objects
local Data,Cols,Sym,Num,Row

-- `Data` is a holder of `rows` and their sumamries (in `cols`).
function Data() return {_is = "Data",
                       cols= nil,  -- summaries of data
                       rows= {}    -- kept data
                       } end

-- `Columns` Holds of summaries of columns.
-- Columns are created once, then may appear in  multiple slots.
function Cols() return {
  _is  = "Cols",
  names={},  -- all column names
  all={},    -- all the columns (including the skipped ones)
  klass=nil, -- the single dependent klass column (if it exists)
  x={},      -- independent columns (that are not skipped)
  y={}       -- depedent columns (that are not skipped)
  } end

-- `Sym`s summarize a stream of symbols.
function Sym(c,s)
  return {_is= "Sym",
         n=0,           -- items seen
         at=c or 0,     -- column position
         name=s or "", -- column name
         _has={}        -- kept data
         } end

-- `Num` ummarizes a stream of numbers.
function Num(c,s)
  return {_is="Nums",
         n=0,at=c or 0, name=s or "", _has={}, -- as per Sym
         isNum=true,      -- mark that this is a number
         lo= math.huge,   -- lowest seen
         hi= -math.huge,  -- highest seen
         isSorted=true,   -- no updates since last sort of data
         w = ((s or ""):find"-$" and -1 or 1)
         } end

-- `Row` holds one record
function Row(t) return {_is="Row",
                       cells=t,            -- one record
                       cooked=copy(t), -- used if we discretize data
                       isEvaled=false    -- true if y-values evaluated.
                       } end

-- ## Data
-- Add one thing to `col`. For Num, keep at most `nums` items.
local 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.isSorted = false
             col._has[pos] = tonumber(v) end end end end

local function adds(col,t) for _,x in pairs(t) do add(col,x) end; return col end

--- Add a `row` to `data`. Calls `add()` to updatie the `cols` with new values.
local 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

--- Generate rows from some `src`.  If `src` is a string, read rows from file;
-- else read rows from a `src`  table. When reading, use rowl to define columns.
local  function records(src,      data,head,body)
  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 -----------
  function body(t) -- treat first row differently (defines the columns)
    if data.cols then record(data,t) else data.cols=head(t) end
  end ---------
  data =  Data()
  if type(src)=="string" then csv(src, body) else
```

```lua
174         for _,t in pairs(src or {}) do body(t) end end
175       return data end
176
177  -- ### Query
178  -- Return kept numbers, sorted.
179  local function nums(num)
180    if not num.isSorted then table.sort(num._has); num.isSorted=true end
181    return num._has end
182
183  -- Diversity (standard deviation for Nums, entropy for Syms)
184  local function div(col)
185    if   col.isNum then local a=nums(col); return (per(a,.9)-per(a,.1))/2.58 else
186      local function fun(p) return p*math.log(p,2) end
187      local e=0
188      for _,n in pairs(col._has) do if n>0 then e=e-fun(n/col.n) end end
189      return e end end
190
191  -- Central tendancy (median for Nums, mode for Syms)
192  local function mid(col)
193    if col.isNum then return per(nums(col),.5) else
194      local most,mode = -1
195      for k,v in pairs(col._has) do if v>most then mode,most=k,v end end
196      return mode end end
197
198       -- Diversity (standard deviation for Nums, entropy for Syms)
199  function div(col)
200    if   col.isNum then local a=nums(col); return (per(a,.9)-per(a,.1))/2.58 else
201      local function fun(p) return p*math.log(p,2) end
202      local e=0
203      for _,n in pairs(col._has) do if n>0 then e=e-fun(n/col.n) end end
204      return e end end
205
206
207  -- For `showCols` (default=`data.cols.x`) in `data`, report `fun` (default=`mid`).
208  local function stats(data,  showCols,fun,    t)
209    showCols, fun = showCols or data.cols.y, fun or mid
210    t={}; for _,col in pairs(showCols) do t[col.name]=fun(col) end; return t end
```

```lua
211  -- --------------------------------
212  local eg, fails = {},0
213
214  local function runs(k)
215    if not eg[k] then return end
216    math.randomseed(the.seed) -- reset seed
217    local old={}; for k,v in pairs(the) do old[k]=v end
218    local out=eg[k]()
219    for k,v in pairs(old) do the[k]=v end -- restore old settings
220    print("!!!!!!", k, out and "PASS" or "FAIL")  end
221
222
223  function eg.LIST(   t)
224    t={}; for k,_ in pairs(eg) do t[1+#t]=k end; table.sort(t); return t end
225
226  function eg.LS()
227    print("\nExamples lua csv -e ...")
228    for _,k in pairs(eg.LIST()) do print(string.format("\t%s",k)) end
229    return true end
230
231  function eg.ALL()
232    for _,k in pairs(eg.LIST()) do
233      if k ~= "ALL" then
234        print"\n------------------------------------"
235        fails = fails + (runs(k) and 0 or 1) end end
236    return true end
237
238  -- Settings come from big string top of "sam.lua"
239  -- (maybe updated from comamnd line)
240  function eg.the() oo(the); return true end
241
242  -- The middle and diversity of a set of symbols is called "mode"
243  -- and "entropy" (and the latter is zero when all the symbols
244  -- are the same).
245  function eg.ent(  sym,ent)
246    sym= adds(Sym(), {"a","a","a","a","b","b","c"})
247    ent= div(sym)
248    print(ent,mid(sym))
249    return 1.37 <= ent and ent <=1.38 end
250
251  -- The middle and diversity of a set of numbers is called "median"
252  -- and "standard deviation" (and the latter is zero when all the nums
253  -- are the same).
254  function eg.num(  num)
255    num=Num()
256    for i=1,100 do add(num,i) end
257    local med,ent = mid(num), div(num)
258    print(mid(num) ,div(num))
259    return 50<= med and med<= 52 and 30.5 <ent and ent <32 end
260
261  -- Nums store only a sample of the numbers added to it (and that storage
262  -- is done such that the kept numbers span the range of inputs).
263  function eg.bignum(  num)
264    num=Num()
265    the.nums = 32
266    for i=1,1000 do add(num,i) end
267    oo(nums(num))
268    return 32==#num._has; end
269
270  -- Show we can read csv files.
271  function eg.csv()
272    csv("../data/auto93.csv",print); return true end
273
274  -- Print some stats on columns.
275  function eg.stats()
276    oo(stats(records("../data/auto93.csv"))); return true end
277
278  -- --------------------------------
279  the = cli(the)
280  runs(the.eg)
281  rogues()
282  os.exit(fails)
```