```
require"lib"

-------------------------------------------------------------------------------
-- config
the={ min  = .5,
      bins = 16,
      some = 256,
      seed = 10019,
      file = "../../data/auto93.csv"}

-------------------------------------------------------------------------------
-- data model
function goalp(x)  return (x or ""):find"[!+-]$" end
function nump(x)   return (x or ""):find"^[A-Z]" end
function assp(x)   return (x or ""):find"!$"  end
function skip(x)   return (x or ""):find":$"  end
function weight(x) return (x or ""):find"-$" and -1 or 1 end

-------------------------------------------------------------------------------
-- col create
function col(at,txt)
  return {n   = 0,
          at  = at  or 0,
          txt = txt or "",
          w   = weight(txt),
          ok  = false,
          log = {},
          div = 0,
          mid = 0
          } end

function num(at,txt)
  local i = col(at,txt)
  i.nump  = true
  i.w     = i.txt:find"-$" and -1 or 1
  i.lo    =  big
  i.hi    = -big
  return i end

-- col update
function cell(i,v,r)
 r = r or 1
 if   v ~= "?"
 then i.n = i.n + r
      if i.nump
      then i.lo = math.min(v, i.lo)
           i.hi = math.max(v, i.hi)
           if     #i.log < the.some     then i.ok=false; push(i.log,v)
           elseif R()    < the.some/i.n then i.ok=false; i.log[ R(#i.log) ]=v end
      else r = r or 1
           i.ok = false
           i.log[v] = r + (i.log[v] or 0) end end
  return i end

function ok(i)
  if not i.ok then
    i.div, i.mid = 0, 0
    if   i.nump
    then i.log = sort(i.log)
         i.mid = per(i.log, .5)
         i.div = (per(i.log, .9) - per(i.log, .1)) / 2.56
    else local most = -1
         for x,n in pairs(i.log) do if v>0 then
           if n > most then most, i.mid = n, x end
           i.div = i.div - n/i.n * math.log( n/i.n, 2) end end end end
  i.ok = true
  return i end

-- col query
function norm(i,x)
  return i.hi - i.lo < 1E-9 and 0 or (x-i.lo)/(i.hi-i.lo) end

-------------------------------------------------------------------------------
-- data create
function data(names)
  local i={x={}, y={}, xy={}, names=names,klass=nil}
  for at,txt in pairs(names) do
    local new = txt:find"^[A-Z]" and num(at,txt) or col(at,txt)
    if not skip(txt) then
      push(goalp(txt) and i.y or i.x, new)
      if klassp(txt) then i.klass=new end end end
  return i end

function rows(src)
  local i
  if type(src)=="table" then for  _,t in pairs(src) do i=row(t,i) end
                        else for    t in csv(src)   do i=row(t,i) end end
  return i end

function clone(i,inits,   j)
  j=row(i.names); for _,t in pairs(inits or {}) do  j=row(t,j) end; return j end

function row(t,i)
  if not i then return data(t) end
  push(i.xy, t)
  for _,cols in pairs{i.x, i.y} do
    for _,c in pairs(cols) do cell(c, t[c.at]) end end end

-- data query
function div(i)     if not i.ok then ok(i) end; return i.div end
function mid(i)     if not i.ok then ok(i) end; return i.mid end
function mids(i, t) t={};for _,c in pairs(i.y) do t[c.txt]=mid(c)end;return t end
function divs(i, t) t={};for _,c in pairs(i.y) do t[c.txt]=div(c)end;return t end
function bin(i,x)
  if   i.nump
  then b=(i.hi - i.lo)/the.bins; return i.lo==i.hi and 1 or math.floor(x/b+.5)*b
  else return x end end
-------------------------------------------------------------------------------
-- row sort
function orders(i,t)
  local function first(t1,t2)
    local s1, s2, n, e = 0, 0, #i.y, math.exp(1)
    for _,c in pairs(i.y) do
      local x,y = norm(c, t1[c.at]), norm(c, t2[c.at])
      s1 = s1 - e^(c.w * (x-y)/n)
      s2 = s2 - e^(c.w * (y-x)/n) end
    return s1/n < s2/n
  end ----------------
  return sort(t, first) end

-------------------------------------------------------------------------------
-- discretization

function ranges(listOfRows,xcol,yklass,y)
  local n,list, dict = 0,{}, {}
  for label,rows in pairs(listOfRows) do
    for _,row in pairs(rows) do
      local v = row[xcol.at]
      if v ~= "?" then
        n = n + 1
        local pos = bin(v)
        dict[pos] = dict[pos] or push(list, {lo=v,hi=v,ys=col(xcol.at, xcol.txt)})
        dict[pos].lo = math.min(v, dict[pos].lo)
        dict[pos].hi = math.max(v, dict[pos].hi)
        cell(dict[pos].ys, label)  end end  end
    list = sort(list, lt"lo")
    list = xcol.nump and _xpand(_merges(list, n^the.min)) or list
    return {ranges= list,
            div   = sum(list,function(z) return div(z.ys)*z.ys.n/n end)} end

function merge(i,j, min)
  local k = col(i.at, i.txt)
  for x,n in pairs(i.ys.log) do cell(k,x,n) end
  for x,n in pairs(j.ys.log) do cell(k,x,n) end
  if i.n<min or j.n<min or div(k) <= (i.n*div(i) + j.n*div(j)) / k.n then
    return {lo=i.lo, hi=j.hi, ys=k} end end

function _merges(b4, min)
  local j,now = 1,{}
  while j <= #b4 do
    local merged
    if j<#b4 then merged = merge(b4[j], b4[j+1], min) end
    now[#now+1] = merged and merged or b4[j]
    j           = j + (merged and 2 or 1) end-- skip to next and next if we found a me
rge
  return #now == #b4 and now or _merges(now,min) end -- hunt for more merges

function _xpand(ranges)
  for j=2,#ranges do ranges[j].lo = ranges[j-1].hi end
  ranges[1].lo, ranges[#t].hi = -big, big
  return ranges end

-------------------------------------------------------------------------------
-- test suite
go,no = {},{}

function go.the() oo(the) end

function go.rows(  i)
  i=rows(the.file)
  oo(i.x[1]) end

function go.stats()
  oo(summarize(rows(the.file) ))
end

function go.order(  i,t)
  i= rows(the.file)
  t= orders(i, i.xy)
  left = clone(i,splice(i.xy,1,30))
  right= clone(i,splice(i.xy,360))
  print("first",o(mids(left)))
  print("last", o(mids(right)))
  print("all",  o(mids(i)))
  end

math.randomseed(the.seed)
if arg[1]=="-g" and type(go[arg[2]])=="function" then go[arg[2]]() end
```