```lua
require"lib"
-- modules start with an Upper case letter
-- class methods are in Module.UPPERCASE (e.g. Module.NEW for constructors)
-- instance methods are in Module.method(i,...)
-- don't say self, say "i" (shorter)
-- where p-- osible, if looking at two instances, use "i,j"
-- types = int,real, str,tab
--------------------------------------------------------------------------------
local the={ min  = .5,
            bins = 16,
            some = 256,
            seed = 10019,
            file = "../../../data/auto93.csv"}

--------------------------------------------------------------------------------
local Col={}
function Col.GOAL(x)   return (x or ""):find"[!+-]$" end
function Col.NUMP(x)   return (x or ""):find"^[A-Z]" end
function Col.KLASS(x)  return (x or ""):find"!$"  end
function Col.SKIP(x)   return (x or ""):find":$"  end
function Col.WEIGHT(x) return (x or ""):find"-$" and -1 or 1 end

--> .COLS(names:[str]) :COLS -> constructor
function Col.COLS(names)
  local i={x={}, y={}, names=names, klass=nil}
  for at,txt in pairs(names) do
    local new = Col.NUMP(txt) and Col.NUM(at,txt) or Col.NEW(at,txt)
    if not Col.SKIP(txt) then
      push(Col.GOAL(txt) and i.y or i.x, new)
      if Col.KLASS(txt) then i.klass=new end end end
  return i end

--> .NEW(at:?int, txt:?str) :COL -> constructor of numbers
function Col.NEW(at,txt)
  return {n  =0,     at=at or 0, txt=txt or "",
          ok =false, kept={},
          div=0,     mid=0} end

function Col.NUM(at,txt,some)
   i     = Col.NEW(at,txt)
   i.w   = Col.WEIGHT(txt)
   i.nums= some or the.some -- if non-nil the i.nums is a numeric
   return i end

function Col.add(i,v,inc)
 inc = inc or 1
 if   v ~= "?"
 then i.n = i.n + inc
      if i.nums
      then for _=1,inc do
            if     #i.kept < i.nums then i.ok=false;push(i.kept,v)
            elseif R() < i.nums/i.n then i.ok=false;i.kept[R(#i.kept)]=v end end
      else i.ok = false
           i.kept[v] = inc + (i.kept[v] or 0) end end
  return i end

function Col.ok(i)
  if not i.ok then
    i.div, i.mid = 0, 0
    if   i.nums
    then i.kept = sort(i.kept)
         i.mid  = per(i.kept, .5)
         i.div  = (per(i.kept, .9) - per(i.kept, .1)) / 2.56
    else local most = -1
         for x,n in pairs(i.kept) do
           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
  i.ok = true end

function Col.lo(i)   Col.ok(i); return i.kept[1] end
function Col.hi(i)   Col.ok(i); return i.kept[#i.kept] end
function Col.div(i)  Col.ok(i); return i.div end
function Col.mid(i)  Col.ok(i); return i.mid end
function Col.norm(i,x)
  local lo,hi = Col.lo(i), Col.hi(i)
  return hi-lo < 1E-9 and 0 or (x-lo)/(hi-lo) end

function Col.bin(i,x)
  if   i.nums then
    local lo,hi = Col.lo(i), Col.hi(i)
    local b=(hi - lo)/the.bins
    x = lo==hi and 1 or math.floor(x/b+.5)*b end
  return x end
--------------------------------------------------------------------------------
local Row={}
function Row.NEW(of,cells) return {of=of,cells=cells,evaled=false} end

function Row.better(i,j)
  local s1, s2, n = 0, 0, #i.of.y
  for _,c in pairs(i.of.y) do
    local x,y =  i.cells[c.at], j.cells[c.at]
    x,y = Col.norm(c, x), Col.norm(c, y)
    s1  = s1 - 2.7183^(c.w * (x-y)/n)
    s2  = s2 - 2.7183^(c.w * (y-x)/n) end
  return s1/n < s2/n  end

--------------------------------------------------------------------------------
local Data={}
function Data.NEW(t) return {rows={}, cols=Col.COLS(t)} end

function Data.ROWS(src,fun)
  if type(src)=="table" then for  _,t in pairs(src) do fun(t) end
                        else for    t in csv(src)   do fun(t) end end end

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

function Data.add(i,t)
  t = t.cells and t or Row.NEW(i,t)
  push(i.rows, t)
  for _,cols in pairs{i.cols.x, i.cols.y} do
    for _,c in pairs(cols) do Col.add(c, t.cells[c.at]) end end end

function Data.mids(i,cols, t)
  t={}
  for _,c in pairs(cols or i.cols.y) do t[c.txt] = Col.mid(c) end;return t end

--------------------------------------------------------------------------------
local Bin={}
function Bin.new(xlo, xhi, ys) return {lo=xlo, hi=yhi, ys-ys} end
function Bin.add(i,x,y)
  i.lo = math.min(i.lo, x)
  i.hi = math.max(i.hi, x)
  Col.add(i.ys, y) end

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

function Bin.BINS(listOfRows,col,y)
  local n,list, dict = 0,{}, {}
  for label,rows in pairs(listOfRows) do
    for _,row in pairs(rows) do
      local v = row[col.at]
      if v ~= "?" then
        n = n + 1
        local pos = Col.bin(col,v)
        dict[pos] = dict[pos] or push(list, Bin.new(v,v,Col.new(col.at,col.txt)))
        Bin.add(dict[pos], v, label) end end end
    list = sort(list, lt"lo")
    list = col.nums and Bin.MERGES(list, n^the.min) or list
    return {bins= list,
            div = sum(list,function(z) return Col.div(z.ys)*z.ys.n/n end)} end

function Bin.MERGES(b4, min)
  local j,now = 1,{}
  while j <= #b4 do
    local merged = j<#b4 and Bin.merge(b4[j], b4[j+1], min)
    now[#now+1]  = merged or b4[j]
    j            = j + (merged and 2 or 1)  end
  if   #now < #b4
  then return Bin.MERGES(now,min) -- loop to look for other merges
  else -- stretch the bins to cover minus infinity to plus Infinity
       for j=2,#now do now[j].lo = now[j-1].hi end
       now[1].lo, now[#now].hi = -big, big
       return now end end

--------------------------------------------------------------------------------
Go,No = {},{}

function Go.THE() oo(the) end

function Go.ROWS(  d)
  Data.ROWS(the.file,function(row)
    if not d then d=Data.NEW(row) else
       Data.add(d,row) end end)
  oo(Data.mids(d)) 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
```