```lua
local b4={}; for k,_ in pairs(_ENV) do b4[k]=k end
local big = 1E34
local tiny= 1/big

local function atom(x)
  if type(x)~="string" then return x end
  x = x:match"^%s*(.-)%s*$"
  if x=="true" then return true elseif x=="false" then return false end
  return tonumber(x) or x end

local function cli(key,x)
  for n,y in pairs(arg) do if y==k then
    x=x=="false" and"true" or x=="true" and"false" or arg[n+1] end end
  return atom(x) end

local function settings() return {
  cohen = cli("-c", .35),
  best  = cli("-b", .85),
  data  = cli("-d", "etc/data/auto93.csv"),
  seed  = cli("-s", 10019)} end

local function atoms(x,  t)
  t={}; for y in x:gmatch(sep or"([^,]+)") do t[1+#t]=atom(y) end; return t end

local function rows(file,      x,prep)
  file = io.input(file)
  return function()
    x=io.read(); if x then return atoms(x) else io.close(file) end end end

as=setmetatable
local function obj(   t)
  t={}; t.__index=t
  return as(t, {__call=function(_,...) return t.new(...) end}) end

-------------------------------------------------------------------------
local Num,Sym,Cols,Data=obj(),obj(),obj(),obj()

local function col(at,x,  i)
  i = {n=0, at=at or 0, txt=txt or "", has={}}
  i.w = i.txt:find"-$" and -1 or 1
  return i end

local function add(self,x,inc)
  if x~="?" then
    inc = inc or 1
    self.n = self.n+1
    self:add1(x,inc or inc) end
  return self end

function Num:new(at,x,   new)
  new = as(col(at,t),self)
  new.mu, new.m2, new.lo, new.hi= 0,0,-big,big
  return new end

function Num:add1(self,x,_,    d)
  d = x - self.mu
  self.mu = self.mu + d/self.n
  self.m2 = self.m2 + d*(x - self.mu)
  self.sd = (self.n<2 or self.m2<0) and 0 or (self.m2/(self.n-1))^.5
  if x > self.max then self.max = x end
  if x < self.min then self.min = x end end

function Num:norm(x)
  return self.hi-self.lo<tiny and 0 or (x-self.lo)/(self.hi-self.lo) end

function Num:heaven(x,   heaven)
  heaven = self.w>0 and 1 or 0
  return (heaven - self:norm(x))^the.p end

function Sym:new(at,x,inc,   new)
  new=as(col(at,x),self); new.most=0; return new end

function Sym:add1(x,inc)
  i.has[x] = inc + (i.has[x] or 0)
  if i.has[x] > i.most then i.most,i.mode=i.has[x],x end end

function Data:new(inits,  new)
  new = as({rows={},heavens=Num()},self)
  if type(inits)=="string" then for   row in csv(inits)   do new:add(row) end end
  if type(inits)=="table"  then for _,row in pairs(inits) do new:add(row) end end

  return new end

function Data:add(t, n)
  if self.cols then self:addData(t) else
     self.cols = Cols(t)
     self.best = self.cols:clone()
     self.rest = self.cols:clone() end end

function Data:addData(t,   n)
  self.rows[1+#self.rows] = self.cols:add(t)
  n = self.heavens.norm( self.heavens.add(self.heaven(t)))
  (n>=the.best and self.best or self.rest):add(t) end

function Data:heaven(t)
  heaven = function(col) return col:heaven(t[col.at]) end
  return (sum(self.cols.y,heaven)/#self.cols.y)^(1/the.p) end

function Cols:new(headers,   new,col,here)
  new = as({all={}, x={}, y={}},self)
  for at,x in pair(headers) do
    if x:find":$" then new.all[n] = Skip(at,x) else
      col = (x:find"^[A-Z]" and Num or Sym)(at,x)
      self.all[at] = col
      here =  x:find"[+-]$" and self.y or self.x
      here[1+#here] = new end end
  return new end

function Cols:add(t)
  for _,col in pairs(self.all) do col:add(t[col.at]) end
  return t end

function Cols:clone(rows,   new)
  new = new or Cols(map(self.cols.all, function(x) return x.txt end))
  for _,row in pairs(rows or {}) do new:add(row) end
  return {rows=rows,cols=new} end

function csv(i,file,   new,about,rows)
  new=new or Cols(about)
  rows={}
  for row in rows(file) do
    if about then rows[1+#rows]=cols1(about,row) else about=cols(row) end end
  return {rows=rows,cols=about} end

as={sym={add=sym1},
    num={add=num1}}

function add(i,x, inc)
  if x ~= "?" then
    inc=inc or 1
    i.n = i.n+inc
    as[i.as].add(i,x,inc) end
  return x end

function what(data, row)
  for _,col in pairs(data.cols.y) do


function main(file, rows,it)
  for row in csv(file) do
    if cols then
      cols
for k,v in pairs(_ENV) do if not b4[k] then print("?",k,type(v)) end end
 --
--
-- NUM={is=NUM}
-- local is={}
-- function is.ignorep(x) return x:find":$" end     -- columns to ignore
-- function is.klassp(x)  return x:find"!$" end     -- symbolic goals to achieve
-- function is.lessp(x)   return x:find"-$" end     -- number goals to minimize
-- function is.morep(x)   return x:find"+$" end     -- numeric goals to maximize
-- function is.nump(x)    return x:find"^[A-Z]" end -- numeric columns
-- function is.goalp(x)   return morep(x) or lessp(x) or klassp(x) end
--
-- function col(k,at,txt,t,   i)
-- local DATA,NUM,SYM={},{},{}
-- function NUM.new(k,at,txt) return col(k,at,txt,{mu=0,m2=0,sd=0})  end
-- function NUM.heaven(i,x)   return (i.w - i:norm(x))^2 end
-- function NUM.norm(i,x)
--   return (i.hi-i.lo)<1E-9 and 0 or (x - i.lo)/(i.hi - i.lo + 1E-9) end
-- function NUM.add(i,x,    d)
--   if x ~="?" then
--     d    = x - i.mu
--     i.mu = i.mu + d/i.n
--     i.m2 = i.m2 + d*(x - i.mu)
--     i.sd = Num.sd0(i)
--     if x > i.max then i.max = x end
--     if x < i.min then i.min = x end end end
--
-- function SYM.new(k,at,txt) return col(k,at,txt,{}) end
-- function SYM.add(i,x)      i.all[x] = 1 + (i.all[x] or 0) end
--
-- function DATA.new(k,t)
--   return new(k,{rows={},cols={},x={},y={}}) end
--
-- function DATA.dth(i,t)
--   local fun = function(col) return col:heaven(t[col.at]) end
--   return (sum(i.y, fun)/#i.y)^.5 end
--
-- function DATA.add(i,t)
--   for at,name in pairs(t) do
--     what= (is.nump(name) and NUM or SYM)(at,name)
--     if is.ignorep(x) then
--
--
-- local the=settings(help)
-- math.randomseed(the.seed)
-- goals={}
-- for n,word in pairs(row) do
--   if is.goalp(word) then
--     goal[n] = is.less[(word) and -1 or 1 end end
--
-- local it= {names={}, cols={}, nums={},x={}, y={}}
--
-- local DATA={}
-- function DATA:new()
--   return new(k,{rows={}, names={}, cols={}, nums={},x={},y={}})
-- function DATA:load(file)
-- for row in csv(the.data) do
--   if 0==#it.cols then
--     it.names=row
--     for n,x in pairs(has) do
--       col = push(it.cols,{})
--       if not is.ignorep(x) then
--         if is.nump[n] then it.nums[n]=true end
--         push(is.goalp(x) and it.y or it.x, col) end end
--   else
--     for n,col in pairs(it.cols) do
--       if num
--
--
--
-- end
```