```
local b4={}; for k,v in pairs(_ENV) do b4[k]=k end
local the,help = {},[[

TODO: dont fuse on non-numerics

 -c cohen difference in nums    = .35
 -f file  source                = ../../data/auto93.csv
 -g go    action                = help
 -m min   size of small         = .5
 -s seed  random number seed    = 10019]]

local cat,chat,cli,csv,fmt,fuse, fumel,kap,lines,lt,map,new
local obj,order,push,rogues,same,sort,thing,trim,words

function lt(x) return function(a,b) return a[x] < b[x] end end

function trim(x) return  x:match"^%s*(.-)%s*$" end

function thing(x)
  if x=="true" then return true elseif x=="false" then return false
  else return math.tointeger(x) or tonumber(x) or x end  end

function lines(file, fun)
  local file = io.input(file)
  while true do
    local line = io.read()
    if not line then return io.close(file) else fun(line) end end end

function words(s,sep,fun,      t)
   fun = fun or same
   t={};for x in s:gmatch(fmt("([^%s]+)",sep)) do t[1+#t]=fun(x) end; return t end

function csv(file,fun)
  lines(file, function(line) fun(words(line, ",", thing)) end) end

function cli(t)
  for key,x in pairs(t) do
    x = tostring(x)
    for n,flag in ipairs(arg) do
      if   flag=="-"..key:sub(1,1)
      then x = x=="false" and "true" or x=="true" and "false" or arg[n+1] end end
    t[key] = thing(x) end
  return t end

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

fmt=string.format
function same(x) return x end
function map(t,f,  u) u={};for _,x in pairs(t)do u[1+#u]=f(x) end;return u end
function kap(t,f,  u) u={};for k,x in pairs(t)do u[1+#u]=f(k,x)end;return u end
function sort(t,f)    table.sort(t,f); return t end
function push(t,x)    t[1+#t]=x; return x end

function chat(t) print(cat(t)); return t end
function cat(t)
  local function pub(k,v) return (tostring(k)):sub(1,1)~="_" end
  local function key(k,v) if pub(k) then  return fmt(":%s %s",k,v) end end
  local u=  #t>1 and map(t,f or tostring) or sort(kap(t,key))
  return (t._is or "").."{"..table.concat(u," ")..."}" end

local _id = 0
function new(kl,...)
  local x
  _id=_id+1; x=setmetatable({_id=_id},kl);kl.new(x,...); return x end

function obj(name)
  local t = {__tostring=cat,_is=name}; t.__index=t
  return setmetatable(t, {__call=new}) end


-------------------------------------------------------
local function col(self, at,txt)
  self.at   = at or 0                 -- :num   column position
  self.txt  = txt or ""               -- :str   column name
  self.n    = 0                       -- :num   items seen so far
  self.all  = {} end

local SYM = obj"SYM"
function SYM:new(at,txt); self.kept={}; col(self,at,txt) end

function SYM:add(x)
  if x ~= "?" then
    self.n=self.n+1
    self.kept[x] = 1+(self.kept[x] or 0) end end

function SYM:bins(of,rows,    x)
  for _,row in pairs(rows) do
    x = row.raw[self.at]
    if x ~= "?" then
      self.all[x] = self.all[x] or  {at=self,lo=x,hi=x,has=of:clone()} end
    self.all[x].has:add(row) end end

function SYM:ent(    e)
  local function z(p) return  p*math.log(p,2) end
  e=0;for _,n in pairs(self.kept)do if n>0 then e=e-z(n/self.n)end end;return e end

local NUM = obj"NUM"
function NUM:new(at,txt)
  col(self,at,txt)
  self.lo =math.huge; self.hi=-self.lo
  self.mu, self.m2, self.sd = 0,0,0
  self._bins= {}
  self.w = self.txt:find"-$" and -1 or 1  end

function NUM:add(x)
  if x ~= "?" then
    self.n  = self.n+1
    local 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.hi then self.hi = x end
    if x < self.lo then self.lo = x end end end

function NUM:bins(of, rows)
  local function lt(x,y)
    return (x=="?" and -math.huge or x) < (y=="?" and -math.huge or y) end
  local order  = function(a,b) return lt(a.raw[self.at],b.raw[self.at]) end
  local x      = function(k)   return rows[k].raw[self.at] end
  local xis    = function(k,x) rows[k].cooked[self.at] = x; end
  local n,b4   = 1,nil
  local min    = self.n^the.min
  local epsilon= self.sd*the.cohen
  local rows   = sort(rows,order)
  local one    = push(self.all, {at=self, n=0, lo=x(1), hi=x(1), has=of:clone()})
  for j,row in pairs(rows) do
    if x(j) ~= "?" then
      if   #rows-j>min and #one.has.rows>min and x(j)~=x(j+1) and x(j)-one.lo>epsilon
      then one = push(self.all, {at=self,lo=one.hi,hi=x(j),has=of:clone()}) end
      one.hi = x(j)
      one.has:add(row) end end
  end

local is={}
is.skip=  function(x) return x:find":$"     end -- what to ignore
is.klass= function(x) return x:find"!$"     end -- single goal
is.goal=  function(x) return x:find"[!+-]$" end -- dependent column
is.num=   function(x) return x:find"^[A-Z]" end -- NUM or SYM?

local COLS = obj"COLS"
function COLS:new(names)
  self.names = names   -- :[str]       list of known columns names
  self.all   = {}      -- :[NUM|SYM]   all the columns
  self.x     = {}      -- :[NUM|SYM]   list of pointers to just the independent column
s
  self.y     = {}      -- :[NUM|SYM]   list of ponters to just the dependent columns
  self.klass = nil     -- :?(NUM|SYM)  pointer to the klass column, may be nil.
  for at,txt in pairs(names) do
    local col = (is.num(txt) and NUM or SYM)(at,txt)
    push(self.all, col)
    if not is.skip(txt) then
      push(is.goal(txt) and self.y or self.x, col)
      if is.klass(txt) then self.klass = col end end end end

function COLS:add(row)
  for _,cols in pairs{self.x,self.y} do
    for _,col in pairs(cols) do col:add(row.raw[col.at]) end end
  return row end

local ROW = obj"ROW"
function ROW:new(of,cells)
  self.raw   = cells
  self.cooked= cells
  self._of   = of
  self.evaled = false end

local ROWS = obj"ROWS"
function ROWS:new() self.rows={}; self.cols=nil end

function ROWS:clone(src)
  return ROWS():add(self.cols.names):adds(src) end

function ROWS:adds(src)
  if   type(src) == "string"
  then csv(src, function(row) self:add(row) end)
  else for _,row in pairs(src or {}) do self:add(row) end end
  return self end

function ROWS:add(row)
  if   self.cols
  then push(self.rows,  self.cols:add( row.raw and row or ROW(self,row)))
  else self.cols = COLS(row) end
  return self end


local go={}
function go.all()
  local want = function(k,_)if k~="all" then return k end end
  for _,x in pairs(sort(kap(go,want))) do
    math.randomseed(the.seed)
    go[x]() end end

function go.help() print(help) end
function go.the()  chat(the) end
function go.csv()  csv(the.file, function(x) chat(x) end) end
function go.clone(r,s)
  print(the.file)
  r=ROWS():adds(the.file)
  s=r:clone()
  print(s) end
  --chat(s.cols.all[1]) end

local function ents(rows)
  local e = 0
  for _,col in pairs(rows.cols.x) do
    local s = SYM()
    for _,row in pairs(rows.rows) do
      local x = row.cooked[col.at]
      if x ~= "?" then s:add(x) end end
    e = e + s:ent() end
  return e end

local function fuse1(i,j,epsilon)
  if math.abs(i.e - j.e) <= epsilon then
    return {at=i.at, lo= i.lo, hi=j.hi, n=i.n+j.n,
            e=  (i.n*i.e + j.n*j.e)/(i.n + j.n)} end end

function fuse(epsilon, b4)
  local n,now = 1,{}
  while n<=#b4 do
    local merged = n<#b4 and fuse1(b4[n],b4[n+1],epsilon)
    now[#now+1]  = merged or b4[n]
    n            = n + (merged and 2 or 1)
  end
  if #now<#b4 then  return fuse(epsilon,now) end
  now[1].lo = -math.huge
  now[#now].hi =  math.huge
  return now end

function go.rows(r, es,all)
  es = NUM()
  all= {}
  r=ROWS():adds(the.file)
  print(ents(r))
  es:add(ents(r))
  for _,col in pairs(r.cols.x) do col:bins(r,r.rows) end
  for _,col in pairs(r.cols.x) do
    print(col.txt, #col.all)
    local here={}
    all[col.at]=here
    for _,range in pairs(col.all) do
      local e = ents(range.has)
      es:add(e)
      print("\t", range.lo, range.hi, #range.has.rows,e)
      push(here, {at=range.at,lo=range.lo, hi=range.hi, n=#range.has.rows,e=e})  end e
nd
  for x,one in pairs(all) do
    all[x] = fuse(es.sd  * the.cohen, sort(one,lt"lo")) end
  print("----------")
  for x,one in pairs(all) do
    print(one[1].at.txt)
    for _, range in pairs(one) do
      print("\t", range.lo, range.hi, range.n, range.e)  end end end
----ddb-----------------------------------------------------------------
help:gsub("\n[-]%S[%s]+([%S]+)[^\n]+= ([%S]+)",
          function(k,x) the[k]=thing(trim(x)) end)
the=cli(the)
go[the.go]()
rogues()
----ddb-----------------------------------------------------------------
-- 12.348722672624
-- Clndrs       3
--              3       4       207     10.186135496684
--              4       6       87      8.2438268409432
--              6       8       104     7.0495719622724
-- Volume       8
--              68      105     106     9.1032053779412
--              105     141     88      9.1314349037882
--              141     173     31      8.1942729502156
--              173     225     27      5.8317127555027
--              225     262     46      5.7889030186077
--              262     305     23      5.2949444385008
--              305     350     39      4.7986838713161
--              350     455     38      5.4934036254341
-- Model        6
--              70      72      84      9.265662209105
--              72      74      67      8.82291850851560
--              74      76      64      8.67893830502770
--              76      78      64      9.02038042062880
--              78      80      58      9.1595375866950
--              80      82      61      7.87320126236590
-- origin       3
--              1       1       249     10.1865161267350
--              2       2       70      8.53631123697010
--              3       3       79      8.40611414892680
-- ..............
-- Clndrs
--              -inf    4       207     10.186135496684
--              4       6       87      8.2438268409432
--              6       inf     104     7.0495719622724
-- Volume
--              -inf    141     194     9.1160105236862
--              141     173     31      8.1942729502156
--              173     inf     173     5.4417775380802
-- Model
--              -inf    80      337     9.001367193192
--              80      inf     61      7.8732012623659
-- origin
--              -inf    1       249     10.186516126735
--              2       inf     149     8.4672805661283
```