```lua
--- -----------------------------------------------------------------------
---
---      __
---     /\ \
---     \ \ \____    ___   _ __    __
---      \ \ '__`\  / __`\/\`'__\/'__`\
---       \ \ \L\ \/\ \L\ \ \ \//\  __/
---        \ \_,__/\ \____/\ \_\\ \____\
---         \/___/  \/___/  \/_/ \/____/
---
--- -----------------------------------------------------------------------
local help=[[

bore == best or rest
(c) 2022, Tim Menzies, BSD 2-clause license.

USAGE:
  lua bore.lua [OPTIONS]

OPTIONS:
  -Dump         stack dump on error = false
  -Format   S   format string        = %5.2f
  -best     F   best space           = .15
  -cohen    F   Cohen's delta        = .35
  -data     N   data file            = etc/data/auto93.csv
  -furthest F   far                  = .9
  -help         show help            = false
  -seed     I   random seed          = 10019
  -todo     S   start-up action      = nothing
]]
--- -----------------------------------------------------------------------
---       __                  _   _
---      / _|_   _ _ __   ___| |_(_) ___  _ __  ___
---     | |_| | | | '_ \ / __| __| |/ _ \| '_ \/ __|
---     |  _| |_| | | | | (__| |_| | (_) | | | \__ \
---     |_|  \__,_|_| |_|\___|\__|_|\___/|_| |_|___/
---
local b4={}; for k,_ in pairs(_ENV) do b4[k]=k end
local big  = 1E32
local tiny = 1E-32
local the  = {}

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

local function cli(txt,   t)
  t={}
  txt:gsub("\n [-]([^%s]+)[^\n]*%s([^%s]+)",function(key,x)
    for n,flag in ipairs(arg) do
      if flag:sub(1,1)=="-" and key:find("^"..flag:sub(2)..".*") then
        x = x=="false" and true or x=="true" and "false" or arg[n+1] end end
    t[key] = atom(x) end)
  return t end

local fmt = string.format

local function sort(t,f) table.sort(t,f); return t end

local function slots(t, u)
  u={}; for k,v in pairs(t) do l=tostring(k); if l:sub(1,1)~="_" then u[1+#u]=k end end;
  return sort(u) end

local function main(the, help, demos)
  if the.help then print(help) else
    for _,todo in pairs(the.todo=="all" and slots(demos) or {the.todo}) do
      math.randomseed(the.seed)
      if type(demos[todo])=="function" then demos[todo]() end end end
  os.exit(demos.fails) end

local function map(t,f, u)
  u={};for k,v in pairs(t) do u[1+#u]=f(v) end; return u end

local function tablep(t) return type(t)=="table" end

local function o(t, seen)
  seen = seen or {}
  if not tablep(t) then return tostring(t) end
  if seen[t] then return "..." end
  seen[t]=t
  local key=function(k) return fmt(":%s %s",k,o(t[k],seen)) end
  local u= #t>0 and map(t,function(x) return o(x,seen) end) or map(slots(t),key)

  return '{'..table.concat(u," ").."}" end

local function oo(t) print(o(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

local function sum(t,f,    n)
  n=0; for _,v in pairs(t) do n=n+f(v) end; return n end

local function tree(t, seen, pre, txt, v)
  pre, seen = pre or "", seen or {}
  if not tablep(t) then return print(fmt("%s%s",pre,t)) end
  if seen[t]       then return print(fmt("%s...",pre))   end
  seen[t]=t
  for _,k in pairs(slots(t)) do
    v= t[k]
    if   tablep(v)
    then print(fmt("%s%s",      pre,k)); tree(v,seen,pre .. "  ")
    else print(fmt("%s%s = %s",pre,k,v)) end end end
```

```lua
--- -----------------------------------------------------------------------
---
---            _
---       ___| | __ _ ___ ___  ___  ___
---      / __| |/ _` / __/ __|/ _ \/ __|
---     | (__| | (_| \__ \__ \  __/\__ \
---      \___|_|\__,_|___/___/\___||___/

local as=setmetatable
local function obj(   t)
  t={__tostring=o}; t.__index=t
  return as(t, {__call=function(_,...) return t.new(_,...) end}) end
---     ____ ____ _
---     |    |  | |
---     |___ |__| |___

local function col(at,txt,  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 + inc
    self:add(x,inc) end
  return self end
---     _  _ _  _ _  _
---     |\ | |  | |\/|
---     | \| |__| |  |

local Num=obj{}
function Num:new(at,x,  new)
  new = as(col(at,x),Num)
  new.mu, new.m2, new.lo, new.hi = 0, 0, big, -big
  return new end

function Num:add(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.hi then self.hi = x end
  if x < self.lo then self.lo = 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)
  return ((self.w>0 and 1 or 0) - self:norm(x))^the.p end
---     ____ _   _ _  _
---     [__   \_/  |\/|
---     ___]   |   |  |

local Sym=obj{}
function Sym:new(at,x,inc,   new)
  new=as(col(at,x),Sym); new.most=0; return new end

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

function Sym:div()
  local function plogp(n,  p) p=n/self.n; return p*math.log(p,2) end
  return -sum(self.has, plogp) end
---     ____ _  _ _ ___
---     [__  |_/  | |__]
---     ___] | \_ | |

local Skip=obj{}
function Skip:new(at,x)  return as(col(at,x),Skip) end
function Skip:add(x,inc) return x end
---     ____ ____ _    ____
---     |    |  | |    [__
---     |___ |__| |___ ___]

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

local Data=obj{}
function Data:new(inits,  new)
  new = as({rows={},heavens=Num()},Data)
  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
```

```lua
--- -----------------------------------------------------------------------------
-
---
---          _                          
---       __| | ___ _ __ ___   ___  ___ 
---      / _` |/ _ \ '_ ` _ \ / _ \/ __|
---     | (_| |  __/ | | | | | (_) \__ \
---      \__,_|\___|_| |_| |_|\___/|___/
---
local Demos = {fails=0}

local function asserts(test, msg)
  print(test and "PASS: "or "FAIL: ",msg or "")
  if not test then
    Demos.fails = Demos.fails+1
    if the.Dump then assert(test,msg) end end end

function Demos.the()     oo(the) end
function Demos.col()     oo(col(10,"Mpg-")) end
function Demos.num(   n) n=Num();
  for x=1,1000 do add(n,x) end; print(n) end

function Demos.sym(   s)
  s=Sym(); for _,x in pairs{1,1,1,1,2,2,3} do add(s,x) end
  asserts(s:div() - 1.376 < 0.005, "entropy") end

function Demos.cols(  c)
  print(Cols({"Clndrs", "Weight", "Hp:", "Lbs-",
               "Acc+", "Model", "origin", "Mpg+"}))
  end

the = cli(help)
main(the, help, Demos)
```