```
local help = [[
BORE: best or rest. u show me a good loser and i'll show u a loser (c) 2022, Tim Menzies <timm@ieee.org> opensource.org/licenses/Fair
  alias bore="lua bore.lua "
bore [OPTIONS]
OPTIONS:
     -bins
                -b max bins
                                                              = 16
OPTIONS (other):
                -s random number seed = 10019
-f where to find data = ../etc/data/auto93.csv
-d dump stack+exit on error = false
-h show help = false
  --seed
--file
   --dump
                                                              = false
= nothing
                  -g start up action
   --go
local function thing(x)  x = x : match^{n \cdot \%} s^{*}(-)\% s^{*} S^{*} \\  if x = ^{m} true^{*} then return true elseif x == ^{m} false^{*} then return false end
   return math.tointeger(x) or tonumber(x) or x end
local the={} help:gsub("\n ([-][-][(\mathbb{m}]+)[\mathbb{m}]+\mathbb{m}]+\mathbb{m}]+\mathbb{m}, function(f1, k, f2, x) for n, f1ag in ipairs(arg) do if f1ag==f1 or f1ag==f2 then x = x=="false" and "func" or x=="true" and "false" or arg[n+1] end end the [k] = thing(x) end)
local as, atom, csv, map, merge, o, oo, obj, ok, patch, per, push, rows, sort
local _, GO, BIN, NUM, SYM, COLS, ROW, EGS
local R, big, fmt
big = math.huge
R = math.random
fmt = string.format
function csv(src)
  src = io.input(src)
   return function (line, row)
     if not line then io.close(src) else
   row=(); for x in line:gmatch("([^,]+)") do row[[1#*row]=thing(x) end
   return row end end end
function oo(t) print(o(t)) end
function o(t, u)
  if #t>0 then return "{"..table.concat(map(t,tostring),"").."}" else
      u={}; for k,v in pairs(t) do u[1+#u] = fmt(":\%s\%s\",k,v) end
return (t.is or "").."{"..table.concat(sort(u),"").."}" end end
function obj(name, t,new)
  function objinale, triew function new(kl,...)

local x=setmetatable({},kl); kl.new(x,...); return x end t = (_tostring=o, is=name or ""); t.__index=t
   return setmetatable(t, {__call=new}) end
```

```
70 BIN=obi"BIN"
         function _.new(i,t) as(i,{at=0, txt="", lo=big, hi= -big, ys={}},t) end function _.of(i,x) return i.ys.has[x] or 0 end
       | as(i,{at=0, txt="", has={}, bins={}}),t) end | function _.new(i,t) | as(i,{at=0, txt="", has={}}, bins={}}),t) end | function _.add(i,x,n) | if x=="?" then i.has[x]=(n or 1)+(i.has[x] or 0) end end
         function _.addy(i,x,y)
if x=="g" then
  i.bins[x] = i.bins[x] or BIN{at=i.at, txt=i.txt, lo=x, hi=x, ys=SYM()}
i.bins[x] = i.bins[x] ond end
        function _.mid(i, m,x)
    m=0; for y,n in pairs(i.has) do if n>m then m,x=n,y end end; return x end
         function _.div(i, n,e)
           n=0; for k,m in pairs (i.has) do n = n + m end
e=0; for k,m in pairs (i.has) do e = e - m/n*math.log(m/n,2) end
return e,n end
       function _.merge(i, j, k)
k=SYM(at=i.at, txt=i.txt)
for x,n in pairs(i.has) do k:add(x,n) end
for x,n in pairs(j.has) do k:add(x,n) end
return k end
         function _.merged(i,j, k)
            k = i:merge(j)
div1, n1 = i:div()
div2, n2 = j:div()
div2, n2 = j:div()
if k:div() < (div1*n1 + div2*n2) / (n1+n2) then return k end end
function __new(i,t)
function __new(i,t)
state __new(
        function _.norm(i,x) return x=="?" and x or (x-i.lo)/(i.hi - i.lo) end
        function _.add(i,x)
  if x=="?" then return x end
             i.ok = nil
            push(i.all,x)
if x > i.hi then i.hi=x elseif x<i.lo then i.lo=x end end</pre>
function _.addy(i,x,y, gap)
           If x==""" then return x end
gap = (i.hi - i.lo)/the.bins
x = (x - i.lo)/yhe.bins
i.bins[x] = i.bins[x] or BIN(at=i.at, txt=i.txt, lo=x, hi=x+gap, ys=SYM())
i.bins[x], ys:add(y) end
        function _.has(i) i.all=i.ok and i.all or sort(i.all);i.ok=true;return i.all end
      function _.mid(i) return per(i:has(), .5) end
function _.div(i) return (per(i:has(), .9) - per(i.has(), .1)) / 2.56 end
        function merge (b4,
                                                                             a,b,c,j,n,tmp)
            if c then a, j = c, j+1 end end
tmp[#tmp+1] = a
  j = j+1 end
return #tmp==#b4 and tmp or merge(tmp) end
        function patch(t)
  for j=2,#t do t[j].lo = t[j-1].hi end
t[1].lo = -big
t[#t].hi = big
```

page 3

```
Someoby*ROW*

Someoby*ROW*

function __new(i,t) as(i,(cells={}),data={}},t) end

function __new(i,t) as(i,(cells={}),data={}},t)

y = i.data.cols.y

sl, s2, e = 0, 0, math.exp(1)

for __col in pairs(y) do

a = col:norm(i.cells[col.at])

b = col:norm(i.cells[col.at])

sl = si - e^*(col.w * (a - b) / #y)

sl = si - e^*(col.w * (a - b) / #y)

return 1/#y < 2/#y

col.seb**COLS*

function _.new(i,t, col)

as(i, (all={}, x={}, y={}), names={}},t)

for at,txt in pairs(i.names) do

col = push(i.all, (txt:find*/[A-Z]" and NUM or SYM){at=at, txt=txt})

if not txt:find**S then

push(txt:find*s then

push(txt:find*s then

push(txt:find*s then

function _.new(i) i.rows,i.cols={},nil end

function _.file(i,file) for row in csv(file) do i:add(row) end; return i end

function _.file(i,file) for row in csv(file) do i:add(row) end; return i end

function _.dad(i,row)

if i.cols

then row = push(i.rows, row.cells and row or ROW(data=i, cells=row)).cells

for k,col in pairs(i.cols.all) do col:add(row[col.at]) end

return i end

function _.mid(i,cs) return map(cs or i.cols.y, function(c) return c:mid() end) end

function _.div(i,cs) return map(cs or i.cols.y, function(c) return c:mid() end) end

function _.div(i,cs) return map(cs or i.cols.y, function(c) return c:mid() end) end

function _.div(i,cs) return map(cs or i.cols.y, function(c) return c:mid() end) end

function _.div(i,cs) return map(cs or i.cols.y, function(c) return c:mid() end) end

function _.div(i,cs) return map(cs or i.cols.y, function(c) return c:mid() end) end

function _.div(i,cs) return map(cs or i.cols.y, function(c) return c:mid() end) end

function _.div(i,cs) return map(cs or i.cols.y, function(c) return c:mid() end) end

function _.div(i,cs) return map(cs or i.cols.y, function(c) return c:mid() end) end
```