```lua
local b4={}; for k,v in pairs(_ENV) do b4[k]=v end
local any,coerce,csv,fails,fmt,go,lt,many,map,obj,push
local no,o,oo,ok,r,rnd,rnds,sort,sum,the,work1,work
local the,help={},[[
small: explore the world better, explore the world for good.
(c) 2022, Tim Menzies

     .-------.
     | Ba    | Bad <----.  planning= (better - bad)
     |    56 |          |  monitor = (bad - better)
     .-------.------.   |
             | Be   |   v
             |    4 | Better
             .------.

USAGE:
  ./bnb [OPTIONS]

OPTIONS:
  -K       -K  manage low class counts     = 1
  -M       -M  manage low evidence counts  = 2
  -best    -B  best set                    = .5
  -bins    -b  max. number of bins         = 16
  -cohen   -c  cohen                       = .35
  -dump    -d  dump stack+exit on error    = false
  -far     -F  how far to go for far       = .9
  -file    -f  file name                   = ../etc/data/auto93.csv
  -goal    -g  goal                        = recurrence-events
  -help    -h  show help                   = false
  -leaves  -l  number of items in leaves   = .5
  -p       -p  coefficient on distance     = 2
  -rest    -R  rest is -R*best             = 4
  -rnd     -r  rounding numbers            = %5.3f
  -seed    -S  seed                        = 10019
  -some    -s  sample size for distances   = 512
  -todo    -t  start up action             = nothing
  -wait    -w  wait                        = 10]]

------------------------------------------------------------------------------

r = math.random
fmt = string.format
function lt(x)       return function(t,u) return t[x] < u[x] end end
function oo(t)       print(o(t)) end
function push(t,x)   t[1+#t]=x; return x end
function sort(t,f)   table.sort(t,type(f)=="string" and lt(f) or f);return t end
function map(t,f, u) u={};for _,v in pairs(t) do u[1+#u]=f(v) end;return u end
function sum(t,f, u) u=0; for _,v in pairs(t) do u=u+f(v)      end;return u end
function many(a,  u) u={};for j=1,n do push(u,any(a)) end;return u end
function any(a)      return a[r()*(#a)//1] end

function rnds(t,f) return map(t, function(x) return rnd(x,f) end) end
function rnd(x,f)
  return fmt(type(x)=="number" and (x~=x//1 and f or the.rnd) or"%s",x) end

function o(t,    u,one)
  one= function(k,v) return #t>0 and tostring(v) or fmt(":%s %s",k,v) end
  u={}; for k,v in pairs(t) do u[1+#u] = one(k,v) end
  if #t==0 then sort(u) end
  return (t.is or "").."{"..table.concat(u," ").."}" end

function coerce(x)
  x = x:match"^%s*(.-)%s*$"
  if x=="true" then return true elseif x=="false" then return false end
  return math.tointeger(x) or tonumber(x) or x end

function csv(src,      cells)
  function cells(s,  t)
    t={}; for y in s:gmatch("([^,]+)") do t[1+#t]=coerce(y) end; return t end
  src = io.input(src)
  return function(x) x=io.read()
    if x then return cells(x) else io.close(src) end end end

function work1(x,    b4)
  print(x)
  b4={}; for k,v in pairs(the) do b4[k]=v end
  print(the.seed)
  math.randomseed(the.seed)
  if go[x] then go[x]() end
  for k,v in pairs(b4) do the[k]=v end end

function work(   t)
  t={}; for k,_ in pairs(go) do push(t,k) end
  for _,x in pairs(sort(t)) do work1(x) end  end

function obj(name,    t,new,str)
  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


------------------------------------------------------------------------------
local Num=obj"Num"
function Num:new(at,txt)
   self.at  = at or 0
   self.txt = txt or ""
   self.n, self.mu, self.m2 = 0,0,0
   self.w   = self.txt:find"-$" and -1 or 1
   self.lo, self.hi = math.huge, -math.huge end

function Num:add(x,        d)
  if x ~="?" then
    self.n  = self.n + 1
    self.lo = math.min(x, self.lo)
    self.hi = math.max(x, self.hi)
    d       = x - self.mu
    self.mu = self.mu + d/self.n
    self.m2 = self.m2 + d*(x - self.mu) end
  return x end

function Num:mid() return self.mu end
function Num:div() return (self.m2/(self.n - 1))^0.5 end

function Num:norm(x,   lo,hi)
  lo,hi= self.lo, self.hi
  return x=="?" and x or hi-lo < 1E-9 and 0 or (x - lo)/(hi - lo) end

function Num:dist(x,y)
  if     x=="?" and y=="?" then return 1 end
  if     x=="?"            then y = self:norm(y); x = y<.5 and 1 or 0
  elseif y=="?"            then x = self:norm(x); y = x<.5 and 1 or 0
  else x,y = self:norm(x), self:norm(y) end
  return math.abs(x - y) end

------------------------------------------------------------------------------
local Sym=obj"Sym"
function Sym:new(at,txt)
   self.at  = at or 0
   self.txt = txt or ""
   self.n   = 0
   self.has, self.mode, self.most = {},nil,0 end

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

function Sym:mid() return self.mode end
function Sym:div(  e)
  e=0; for _,v in pairs(t) do e=e-v/self.n*log(v/self.n,2) end; return e end

function Sym:dist(x,y) return x=="?" and y=="?" and 1 or x==y and 0 or 1 end

------------------------------------------------------------------------------
local Cols=obj"Cols"
function Cols:new(names,    col)
  self.names, self.all, self.x, self.y, self.klass = names, {}, {}, {}, nil
  for at,txt in pairs(names) do
    col = push(self.all, (txt:find"^[A-Z]" and Num or Sym)(at,txt))
    if not txt:find":$"  then
      if txt:find"!$" then self.klass=col end
      col.indep = not txt:find"[-+!]$"
      push(col.indep and self.x or self.y, col) end end  end

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

------------------------------------------------------------------------------
local Egs=obj"Egs"
function Egs:new() self.rows,self.cols = {}, nil end

function Egs:clone(rows,   out)
  out = Egs():add(self.cols.names)
  for _,row in pairs(rows or {}) do out:add(row) end
  return out end

function Egs:load(file)
  for row in csv(file) do self:add(row) end; return self end

function Egs:add(t)
  if self.cols then push(self.rows, self.cols:add(t)) else self.cols=Cols(t) end
  return self end

function Egs:better(row1,row2)
  local s1, s2, n, e = 0, 0, #self.cols.y, math.exp(1)
  for _,col in pairs(self.cols.y) do
    local a = col:norm(row1[col.at])
    local b = col:norm(row2[col.at])
    s1      = s1 - e^(col.w * (a - b) / n)
    s2      = s2 - e^(col.w * (b - a) / n) end
  return s1 / n < s2 / n  end

function Egs:betters()
  return sort(self.rows, function(a,b) return self:better(a,b) end)  end

function Egs:mid(cols)
  return rnds(map(cols or self.cols.y, function(col) return col:mid() end)) end

function Egs:dist(row1,row2,   d,n)
  d= sum(self.cols.x,function(c) return c:dist(row1[c.at],row2[c.at])^the.p end)
  return (d / (#self.cols.x)) ^ (1/the.p) end
```

```
--------------------------------------------------------------------------------
fails,go,no = 0,{},{}
function ok(test,msg)
  print("", test and "PASS "or "FAIL ", msg or "")
  if not test then
    fails= fails+1
    if  the.dump then assert(test,msg) end end end

function go.eg1(   eg)
  eg = Egs():load(the.file)
  print(eg.cols.y[1]) end

function go.dist(  eg,row2,t)
  eg = Egs():load(the.file)
  t={}; for i=1,20 do
    row2= any(eg.rows)
    push(t, {dist=eg:dist(eg.rows[1],row2), row = row2}) end
  oo(eg.rows[1])
  for _,two in pairs(sort(t,lt"dist")) do oo(two.row) end end

function go.mids( eg,hi,lo,out)
  eg = Egs():load(the.file)
  oo(map(eg.cols.y,function(col) return col.txt end))
  oo(eg:mid())
  lo,hi = eg:clone(), eg:clone()
  for i,row in pairs(eg.rows) do
    if i < 20            then lo:add(row) end
    if i > #eg.rows - 20 then hi:add(row) end end
  oo(lo:mid())
  oo(hi:mid()) end

--------------------------------------------------------------------------------
help:gsub("\n ([-]([^%s]+))[%s]+(-[^%s]+)[^\n]*%s([^%s]+)",
  function(long,key,short,x)
    for n,flag in ipairs(arg) do
      if flag==short or flag==long then
        x = x=="false" and "true" or x=="true" and "false" or arg[n+1] end end
    the[key] = coerce(x) end)

if the.help then print(help) end
if the.todo=="all" then work() else work1(the.todo) end
for k,v in pairs(_ENV) do if not b4[k] then print("?",k,type(v)) end end
os.exit(fails)
```