```lua
-- gate: 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
--                .------.

b4={}; for k,_ in pairs(_ENV) do b4[k]=k end
local r,abs,log,ent,min,max
local sort,slots,copy,push,fmt,fmt2,map,map2,cat,cat2,rnd,rnds
local adds,class,thing,things,csv
local ok,cli,demos,demo
local fails,go,no = 0, {}, {}
local Num,Sym,Cols,Egs

local the={
     k    = 2,
     m    = 1,
     seed = 10019,
     rnd  = "%5.3f",
     dump = false,
     todo = "the",
     keep = 512}

--------------------------------------------------------------------------------
r=    math.random
abs=  math.abs
log=  math.log
min=  math.min
max = math.max

push= function(t,x) t[1 + #t] = x; return x end
sort= function(t,f) table.sort(t,f); return t end

fmt=  string.format
fmt2= function(k,v) return fmt(":%s %s",k,v) end

map=  function(t,f, u) u={};for _,v in pairs(t)do u[1+#u]=f(v)  end;return u end
map2= function(t,f, u) u={};for k,v in pairs(t)do u[1+#u]=f(k,v)end;return u end

copy= function(t,   u)
        if type(t) ~= "table" then return t end
        u={};for k,v in pairs(t) do u[copy(k)]=copy(v) end; return u end

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

cat=  function(t)     return "{"..table.concat(map(t,tostring),   ", ").."}" end
cat2= function(t,sep,    slot)
        function slot(k) return fmt2(k, t[k]) end
        return "{"..table.concat(map(slots(t),slot),sep or " ").."}" end

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

ent= function(t,   n,e)
       n=0; for _,v in pairs(t) do n=n+v end
       e=0; for _,v in pairs(t) do e=e-v/n*log(v/n,2) end; return e end

thing= function(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

things= function(s,sep,   t)
          t={}; for y in s:gmatch("([^,]+)") do t[1+#t]=coerce(y) end
          return t end

csv= function(src)
       src = io.input(src)
       return function(x) x=io.read()
         if x then return things(x) else io.close(src) end end end

class= function(name,    t,new)
         function new(klass,...)
           local obj= setmetatable({},klass)
           local res= klass.new(obj,...)
           if res then obj = setmetatable(res,klass) end
           return obj end
         t={__tostring=cat2, is=name or ""}; t.__index=t
         return setmetatable(t, {__call=new}) end

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

cli= function(the,    k,v)
       for n,flag in ipairs(arg) do
         k = flag:sub(3)
         v = the[k]
         if v ~= nil then
           v = v==false and"true" or v==true and"false" or arg[n+1]
           the[k] = thing(v) end end
       return the end

ok=  function(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

demos= function(the,go,   old,demo1)
         function demo1(txt,fun)
           the = copy(old)
           math.randomseed(the.seed or 10019)
           print(txt)
           fun() end
         ---------
         old = copy(the)
         if   the.todo=="all"
         then for _,txt in pairs(slots(go)) do demo1(txt, go[txt]) end
         else demo1(the.todo, go[the.todo]) end end


```

```lua
--------------------------------------------------------------------------------
Num=class("Num")
function Num:new(at,name)
  self.at, self.name = at or 0, name or ""
  self.w = self.name:find"$-" and -1 or 1
  self.some, self.ok = {}, false
  self.n,self.md,self.sd,self.lo,self.hi = 0,0,0,1E32,-1E32 end

function Num:add(x,_,   a,d)
  if x ~="?" then
    self.n = self.n + 1
    d      = x - self.mu
    self.mu= self.mu + d/self.n
    self.m2= self.m2 + d*(x - self.mu)
    self.sd= (self.m2<0 or self.n<2) and 0 or ((self.m2/(self.n - 1))^0.5)
    self.lo= min(x, self.lo)
    self.hi= max(x, self.hi)
    a      = self.some
    if     #a  < the.num.keep        then self.ok=false; push(a,x)
    elseif r() < the.num.keep/self.n then self.ok=false; a[r(#a)]=x end end
  return x end

function Num:mid() return self.mu end
function Num:div() return self.sd end

function Num:like(x,_)
  local z, e, pi = 1E-64, math.exp(1), math.pi
  if x < self.mu - 4*self.sd then return 0 end
  if x > self.mu + 4*self.sd then return 0 end
  return e^(-(x - self.mu)^2 / (z + 2*self.sd^2))/(z + (pi*2*self.sd^2)^.5) end

--------------------------------------------------------------------------------
Sym=class("Sym")
function Sym:new(at,name)
  self.at, self.name = at or 0, name or ""
  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() return ent(self.has) end

function Sym:like(x,prior)
  return ((self.has[x] or 0) + the.m*prior)/(self.n + the.m) end

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

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

function Egs:add(row,   add)
  add = function(col) col:add(row[col.at]) end
  if self.cols then push(self.rows, map(self.cols,add)) else
     self.cols = Cols(row) end end

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

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

function Egs:like(row,egs,       n,prior,like,col)
  n=0; for _,eg in pairs(egs) do n = n + #eg.rows end
  prior = (#self.rows + the.k) / (n + the.k * #egs)
  like  = log(prior)
  for at,x in pairs(row) do
    col = self.cols.all[at]
    if x ~= "?" and col.indep then like= like + log(col:like(x,prior)) end end
  return like end


```

```lua
------------------------------------------------------------------------------
function go.the() print(cat2(the)) end
function go.aa() print(11) end

the = cli(the)
demos(the,go)

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