```
-- vim: ts=2 sw=2 et:
b4={}; for k,_ in pairs(_ENV) do b4[k]=k end
local help = [[

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
               .------.

OPTIONS (inference control):
  -k     int   Bayes: handle rare classes          = 2
  -m     int   Bayes: handle rare values           = 1
  -seed  int   random number seed                  = 10019
  -keep  int   numbers to keep per column          = 512

OTHER:
  -h           show help                           = false
  -dump        enable stack dump on failures       = false
  -rnd   str   pretty print control for floats     = %5.3f
  -todo  str   start-up action ("all" == run all) = the
]]

local copyright= [[
Copyright (c) 2022 Tim Menzies
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice,
   this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice,
   this list of conditions and the following disclaimer in the documentation
   and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
]]

-------------------------------------------------------------------------------
local the,go,no,fails = {}, {}, {}, 0

local r,abs,log,min,max,ent -- maths
r=    math.random
abs=  math.abs
log=  math.log
min=  math.min
max=  math.max
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

local push,sort,map,map2,copy,slots -- lists
push= function(t,x) t[1 + #t] = x; return x end
sort= function(t,f) table.sort(t,f); return t 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[k] = 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

local fmt,fmt2,o,oo,ooo,rnd,rnds -- printing
fmt=  string.format
fmt2= function(k,v) return fmt(":%s %s",k,v) end

o =  function(t,s) return "{"..table.concat(map(t,tostring),s or", ")..."}" end
oo=  function(t,sep,    slot)
       function slot(k) return fmt2(k, t[k]) end
       return (t.is or"")..o(map(slots(t),slot),sep or" ") end
ooo= function(t) print( #t>1 and o(t) or oo(t)) 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

local coerce,coerces,csv -- read strings/files into lua variables
coerce= 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

coerces= 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 coerces(x) else io.close(src) end end end

local class -- object support
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=oo, is=name or ""}; t.__index=t
         return setmetatable(t, {__call=new}) end

local adds -- misc
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

local cli,ok,demo1,main -- startup and execution
cli= function(help,arg,   t,k,v)
       t={}
       help:gsub("\n [-](%S+)[%s]+[^\n]*%s([^%s]+)",function(k,x) t[k]=x end)
       for n,flag in ipairs(arg) do
         if flag:sub(1,1) == "-" then
           k = flag:sub(2)
           v = the[k]
           assert(v ~= nil,fmt("unknown command line flag [%s]",flag))
           t[k] = v=="false" and "true" or v=="true" and "false" or arg[n+1] end end
       return map2(t,function(k,v) return coerce(v) end) 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

demo1= function(txt,fun,defaults)
         the = copy(defaults)
         math.randomseed(the.seed or 10019)
         print(txt)
         fun() end

main= function(the,go,  defaults)
        the = cli(help,arg)
        if the.h then os.exit(print(help)) end
        defaults = copy(the)
        if   the.todo=="all"
        then for _,txt in pairs(slots(go)) do demo1(txt, go[txt],defaults) end
        else demo1(the.todo, go[the.todo],defaults)
        end
        for k,v in pairs(_ENV) do if not b4[k] then print("?",k,type(v))end end
        os.exit(fails) end

```

```lua
-------------------------------------------------------------------------------
local 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

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
-------------------------------------------------------------------------------
local 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

-------------------------------------------------------------------------------
local 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

-------------------------------------------------------------------------------
local 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

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 go.the() ooo(the) end

main(the,go)
```