```lua
#!/usr/bin/env lua
-- vi: filetype=lua :
-------------------------------------------------------------------------
---  __                __                                  __
---  /\ \              /\ \                              /\ \
---  \ \ \____  _ __   \ \ \/'\      ___      __   __  __\_\ \
---   \ \ '__`\/\`'__\  \ \ , <     /' _ `\  /'__`\/\ \/\ \/'_` \
---    \ \ \L\ \ \ \/    \ \ \\`\   /\ \/\ \/\  __/\ \ \_\ \L\ \
---     \ \_,__/\ \_\     \ \_\ \_\ \ \_\ \_\ \____\\ \____,_\
---      \/___/  \/_/      \/_/\/_/  \/_/\/_/\/____/ \/___,_ /

-- BSD 2-Clause License
-- Copyright (c) 2022, Tim Menzies
--
-- 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 & 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 b4={}; for k,_ in pairs(_ENV) do b4[k]=k end
local help=[[
brknbad.lua: 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:
  -bins  -b   max. number of bins            = 16
  -cohen -c   cohen                          = .35
  -goal  -g   goal                           = recurrence-events
  -K     -K   manage low class counts        = 1
  -M     -M   manage low evidence counts     = 2
  -seed  -S   seed                           = 10019
  -wait  -w   wait                           = 10

OPTIONS (other):
  -dump  -d   dump stack on error, then exit = false
  -file  -f   file name                      = ../etc/data/breastcancer.csv
  -help  -h   show help                      = false
  -todo  -t   start up action                = nothing
]]

local ent,per
local push,map,collect,copy,powerset
local sort,up1,upx,down1,slots,up1,down1
local words,thing, things, items
local cli
local fmt,o,oo
local inc,inc2,inc3,has,has2,has3
local ok,ish, rogues
local cols,update,classify,test,train,score,nb1,nb2,abcd
local bins,nb3
local eg,the,ako={},{},{}

---  _ _  |  _  _  _  _  _  _|_
---  (_(_) | (_)| )(_)| )(_| |_\/|_)(/__\
---                     /     |

local ako={}
ako.num     = function(x) return x:find"^[A-Z]" end
ako.goal    = function(x) return x:find"[-+!]"  end
ako.klass   = function(x) return x:find"!$"     end
ako.ignore  = function(x) return x:find":$"     end
ako.weight  = function(x) return x:find"-$" and -1 and 1 end

---  __|_ _  __|_ _
---  _\ | |_|(_ | _\

local it={}
function it.num()
  return {nump=true,  n=0, at=0, txt="",lo=1E32, hi=-1E32, mu=0, bins={}} end

function it.sym()
  return {nump=false, n=0, at=0, txt="", has={}, most=0, mode=nil} end

function it.three()
  return {all={}, nums={}, syms={}} end

function it.cols()
  return {names={}, klass=nil,xy= it.three(), x= it.three(), y= it.three()} end

function it.egs()
  return {h={}, nh=0, e={}, ames=nil, n=0, bests=0, rests=0,
          best={}, rest={}, log={}, cols=nil} end

-------------------------------------------------------------------------
---  ___ ___ ___ ___ ___
---  |__] |--| [__  |  |__
---  |__] |  | ___] |  |__

function classify(i,t)
  local hi,out = -1
  for h,_ in pairs(i.h) do
    local prior = ((i.h[h] or 0) + the.K)/(i.n + the.K*i.nh)
    local l = prior
    for col,x in pairs(t) do
      if x ~= "?" and col ~= #t then
        l=l*(has3(i.e,col,x,h) + the.M*prior)/((i.h[h] or 0) + the.M) end end
    if l>hi then hi,out=l,h end end
  return out end

function test(i,t)
  if i.n > the.wait then push(i.log,{want=t[#t], got=classify(i,t)}) end  end

function train(i,t)
  local more, kl = false, t[#t]
  for col,x in pairs(t) do
    if x ~="?" then
      more = true
      inc3(i.e, col, x, kl)
      if col ~= #t then
        inc2(kl==the.goal and i.best or i.rest, col,x) end end end
  if more then
    i.n = i.n + 1
    if not i.h[kl] then i.nh = i.nh + 1 end
    inc(i.h, kl)
    if kl==the.goal then i.bests=i.bests+1 else i.rests=i.rests+1 end end end

function score(i)
  local acc,out=0,{}
  for _,x in pairs(i.log) do if x.want==x.got then acc=acc+1/#i.log end end
  for col,xns in pairs(i.best) do
    for x,b in pairs(xns) do
      local r1 = has2(i.rest,col,x)/i.rests
      local b1 = b/i.bests
      push(out, {100*(b1^2/(b1+r1))//1, col,x,b}) end end
  return acc, sort(out,down1) end

function nb1(file, log)
  local i = {h={}, nh=0,e={}, names=nil, n=0, wait=the.wait,
             bests=0,rests=0,best={}, rest={},log=log or {}}
  for row in items(file) do
    if not i.names then i.names=row else
      test(i,row); train(i,row) end end
  return i end

---  _  _ _ _ _  _ _  _    _ _ _|_ _|_
---  |_||_|| | |_\    (_||_|(_|    _\\/|_|_\
---                                  /

function cols(names)
  local i = it.cols()
  local function keep(now, at)  -- keep in "all" plus in one of "nums" or "syms"
    push(ako.num(now.txt) and at.nums or at.syms, push(at.all, now)) end
  i.names = names
  for j,txt in pairs(names) do
    local now = ako.num(txt) and it.num() or it.sym()
    now.at, now.txt, now.w = j, txt, ako.weight(txt)
    keep(now, i.xy)
    if not ako.ignore(txt) then
      keep(now, ako.goal(txt) and i.y or i.x)
      if ako.klass(txt) then i.klass=now end end end
  return i end

function update(i,t)
  local function num(col, x)
    col.mu = col.mu + (x - col.mu)/col.n
    col.lo = math.min(x, col.lo)
    col.hi = math.max(x, col.hi)  end

  local function num(col, x)
    col.has[x] = 1 + (col.has[x] or 0)
    if col.has[x] > col.most then col.mode,col.most = x,col.has[x] end end

  for _,col in pairs(i.cols.xy.all) do
    local x = t[col.at]
    if x ~= "?" then
      col.n = col.n + 1
      (col.nump and num or sym)(col,x) end end
  return t end

---  vv | -|- |-|   _  _|
---  vv | -|- |-|   (/_ vv (_|

function nb2(file,  log)
  local tmp, i = {}, it.egs()
  local function discretize(j,x)
    if x~="?" then
      col = i.cols.xy.all[j]
      if col.nump then
        x = (x - col.lo) // ((col.hi - col.lo+1E-32) / the.bins) end end
    return x end
  -- start
  for row in items(file) do
    if not i.cols then i.cols=cols(row) else push(tmp,update(i,row)) end end
  for _,row in pairs(tmp) do
    row=collect(row,discretize)
    test(i,row); train(i,row) end
  return i end
```

```
--------------------------------------------------------------------------------
---    _  _  ____  ____  ___   _  ____  ____
---   |\/| |___    |   |__/ |  |     [__
---   |  | |___    |   |  \ |  |___  ___]

function abcd(gotwants, show)
  local i, exists, add, report, pretty
  i={data=data or "data",rx= rx or "rx",known={},a={},b={},c={},d={},yes=0,no=0}

  function exists(x,   new)
    new = not i.known[x]
    inc(i.known,x)
    if new then
      i.a[x]=i.yes + i.no; i.b[x]=0; i.c[x]=0; i.d[x]=0 end end

  function report(    p,out,a,b,c,d,pd,pf,pn,f,acc,g,prec)
    p = function (z) return math.floor(100*z + 0.5) end
    out= {}
    for x,_ in pairs( i.known ) do
      pd,pf,pn,prec,g,f,acc = 0,0,0,0,0,0,0
      a= (i.a[x] or 0); b= (i.b[x] or 0); c= (i.c[x] or 0); d= (i.d[x] or 0);
      if b+d > 0     then pd   = d     / (b+d)        end
      if a+c > 0     then pf   = c     / (a+c)        end
      if a+c > 0     then pn   = (b+d) / (a+c)        end
      if c+d > 0     then prec = d     / (c+d)        end
      if 1-pf+pd > 0 then g=2*(1-pf) * pd / (1-pf+pd) end
      if prec+pd > 0 then f=2*prec*pd / (prec + pd)   end
      if i.yes + i.no > 0 then
         acc= i.yes / (i.yes + i.no) end
      out[x] = {data=i.data,rx=i.rx,num=i.yes+i.no,a=a,b=b,c=c,d=d,acc=p(acc),
                prec=p(prec), pd=p(pd), pf=p(pf),f=p(f), g=p(g), class=x} end
    return out end

  function pretty(t)
    print""
    local s1  = "%10s | %10s | %4s | %4s | %4s | %4s "
    local s2  = "| %3s | %3s| %3s | %4s | %3s | %3s |"
    local d,s = "---", (s1 .. s2)
    print(fmt(s,"db","rx","a","b","c","d","acc","pd","pf","prec","f","g"))
    print(fmt(s,d,d,d,d,d,d,d,d,d,d,d,d))
    for _,x in pairs(slots(t)) do
      local u = t[x]
      print(fmt(s.." %s", u.data,u.rx,u.a, u.b, u.c, u.d,
                          u.acc, u.pd, u.pf, u.prec, u.f, u.g, x)) end end
  -- start
  for _,one in pairs(gotwants) do
    exists(one.want)
    exists(one.got)
    if one.want == one.got then i.yes=i.yes+1 else i.no=i.no+1 end
    for x,_ in pairs(i.known) do
      if   one.want == x
      then inc(one.want == one.got and i.d or i.b, x)
      else inc(one.got  == x       and i.c or i.a, x) end end end
  return show and pretty(report()) or report() end
```

```
--------------------------------------------------------------------------------
---   ____ _  _ ___  ____ ____   ____ ____ _  _ ____ ____ ____
---   [__  |  | |__] |___ |__/   |__/ |__| |\ | | __ |___ [__
---   ___] |__| |    |___ |  \   |  \ |  | | \| |__] |___ ___]

function nb3(file,  log)
  local tmp, i = {}, it.egs()
  local function discretize(j,x,   bins)
    if x ~= "?" then
      bins = i.nums[j]
      if bins then
        for _,bin in pairs(bins) do
          if bin.lo <= x and x < bin.hi then return bin.id end end end end
     return x end

  function update1(i, row)
    update(i, row)
  -- start
  for row in items(file) do
    if not i.cols then i.cols = cols(row) else push(tmp,update1(i,row)) end end
  for _,col in pairs(i.cols.x.nums) do i.nums[j] = bins(xys,j) end
  for _,row in pairs(tmp) do
    row = collect(row, discretize);
    test(i,row); train(i,row) end
  return i end

---    |` .  _  _|   |_  .  _  _
---   ~|~ | | |(_|   |_) | | |_\

function bins(xys,ref)
  xys   = sort(xys, upx)
  local cohen    = the.cohen * (per(xys,.9).x - per(xys, .1).x) / 2.56
  local minItems = #xys / the.bins
  local out, b4  = {}, -math.huge
  local function add(f,z) f[z] = (f[z] or 0) + 1 end
  local function sub(f,z) f[z] =  f[z] - 1       end
  local function argmin(lo,hi)
    local lhs, rhs, cut, div, xpect, xy = {},{}
    for j=lo,hi do add(rhs, xys[j].y) end
    div = ent(rhs)
    if hi-lo+1 > 2*minItems
    then
      for j=lo,hi - minItems do
        add(lhs, xys[j].y)
        sub(rhs, xys[j].y)
        local n1,n2 = j - lo +1, hi-j
        if   n1         > minItems and          -- enough items (on left)
             n2         > minItems and          -- enough items (on right)
             xys[j].x ~= xys[j+1].x and         -- there is a break here
             xys[j].x  - xys[lo].x > cohen and -- not trivially small (on left)
             xys[hi].x - xys[j].x  > cohen     -- not trivially small (on right)
        then xpect = (n1*ent(lhs) + n2*ent(rhs)) / (n1+n2)
             if xpect < div then                -- cutting here simplifies things
               cut, div = j, xpect end end end --end for
    end -- end if
    if   cut
    then argmin(lo,    cut)
         argmin(cut+1, hi )
    else b4 = push(out,{ref=ref,lo=b4, hi=xys[hi].x, n=hi-lo+1, div=div}).hi end
  end ----------------------------------------------
  argmin(1,#xys)
  for j,bin in pairs(out) do bin.id = j end
  out[#out].hi =  math.huge
  return out end
```

```
--------------------------------------------------------------------------------
---
---
---
--------------------------------------------------------------------------------
---
---
---

---
---

function per(t,p) return t[ (p or .5)*#t//1 ] end

function ent(t)
  local n=0; for _,m in pairs(t) do n = n+m end
  local e=0; for _,m in pairs(t) do if m>0 then e= e+m/n*math.log(m/n,2) end end
  return -e end

---
---

function ish(x,y,z) return math.abs(x-y) <= (z or 0.001) end

local fails=0
function ok(test,msg)
  print("", test and "PASS "or "FAIL ",msg or "")
  if not test then
    fails = fails+1
    if the and the.dump then assert(test,msg) end end end

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

---
---

function inc(f,a,n)      f=f or{};f[a]=(f[a] or 0) + (n or 1) return f end
function inc2(f,a,b,n)   f=f or{};f[a]=inc( f[a] or {},b,n);  return f end
function inc3(f,a,b,c,n) f=f or{};f[a]=inc2(f[a] or {},b,c,n);return f end

function has(f,a)      return f[a]                    or 0 end
function has2(f,a,b)   return f[a] and has( f[a],b)   or 0 end
function has3(f,a,b,c) return f[a] and has2(f[a],b,c) or 0 end

---
---

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

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

function powerset(s)
  local function aux(s)
    local t = {{}}
    for i = 1, #s do
      for j = 1, #t do
        t[#t+1] = {s[i],table.unpack(t[j])} end end
    return t end
  return sort(aux(s), function(a,b) return #a < #b end) end

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

function upx(a,b)   return a.x < b.x end
function up1(a,b)   return a[1] < b[1] end
function down1(a,b) return a[1] > b[1] end


function slots(t, u)
  local 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

---
---
---

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

function things(s) return map(words(s), thing) end

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

function items(src,f)
  local function file()
    src,f = io.input(src),f or things
    return function() x=io.read();if x then return f(x) else io.close(src) end e
nd end
  local function tbl(   x)
    x,f = 0, f or function(z) return z end
    return function() if x< #src then x=x+1; return f(src[x]) end end end
  return type(src) == "string" and file() or tbl() end

---
---
---

fmt = string.format

function oo(t) print(o(t)) end

function o(t,  seen, u)
  if type(t)~="table" then return tostring(t) end
  seen = seen or {}
  if seen[t] then return "..." end
  seen[t] = t
  local function show1(x) return o(x, seen) end
  local function show2(k) return fmt(":%s %s",k, o(t[k],seen)) end
  u = #t>0 and map(t,show1) or map(slots(t),show2)
  return (t.s or "").."{"..table.concat(u," ").."}" end

---
---

function cli(help)
  local d,used = {},{}
  help:gsub("\n  ([-])([^%s]+))[%s]+([^%s]+)[^\n]*%s([^%s]+)",
    function(long,key,short,x)
      assert(not used[short], "repeated short flag ["..short.."]")
      used[short]=short
      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
      d[key] = x==true and true or thing(x) end)
  if d.help then os.exit(print(help)) end
  return d end
```

```
--------------------------------------------------------------------------------
---
---    ___ ___ _ _ ___ ___
---    |__ \ |__ | \/ | |__| |__]
---
function eg.copy(     t,u)
  t={a={b={c=10},d={e=200}}, f=300}
  u= copy(t)
  t.a.b.c= 20
  print(u.a.b.c)
  oo(t)
  oo(u)
  end

function eg.collect()
  local function aux(x,y) return x*y end
  oo(collect({10,20,30},aux)) end

function eg.ent()
  ok(ish(ent{a=9,b=7}, .98886), "entropy")  end

function eg.items()
  for  x in items{10,20,30} do print(x) end
  local n=0
  for  x in items(the.file) do n=n+1; if n<=5 then oo(x) end end end

function eg.powerset()
  for _,x in pairs(powerset{10,20,30,40,50}) do oo(x) end end

function eg.nb1()
  local i = nb1(the.file);
  local acc, out = score(i); print(acc); map(out,oo) end

-- there is a "?"in the output. nope
function eg.nb2()
  local i = nb2(the.file);
  local acc, out = score(i); print(acc); map(out,oo) end

function eg.nb2a()
  local i = nb2("../etc/data/diabetes.csv");
  local acc, out = score(i)
  abcd(i.log, true)
  map(out,oo) end

function eg.bins(   t)
  local t,n = {},30
  for j=1,n do push(t, {x=j, y=j<.6*n and 1 or j<.8*n and 2 or 3}) end
  map(bins(t,20),oo)
end

function eg.nb3(  i)
  print(20)
  i=nb3("../etc/data/diabetes.csv")
  for n,bins in pairs(i.nums) do
    print(n,#bins) end
  local acc, out = score(i)  -- XXX
  print(#out)
  print(acc)
  map(out,oo)
  end

--------------------------------------------------------------------------------
---
---    ___ ___ ___ ___ ___
---    [__  |  |__| |__/ |
---    ___] |  |  | |  \ |
---
fails = 0
local defaults=cli(help)
local todos = defaults.todo == "all" and slots(eg) or {defaults.todo}
for _,todo in pairs(todos) do
  the = copy(defaults)
  math.randomseed(the.seed or 10019)
  if eg[todo] then eg[todo]() end end

rogues()
os.exit(fails)

---              .---------.
---              |         |
---           -=  _________  =-
---              |  ___ ___  |
---              | |   )=(  | |
---              |  ---   ---  |
---                    ###
---                #   =   #             "This ain't chemistry.
---                #######               This is art."
```