```
---    __                  __                  __                      __
---   /\ \                /\ \                /\ \                    /\ \
---   \ \ \____   _ __    \ \ \/'\     ___    \ \ \____     __       \_\ \
---    \ \ '__`\ /\`'__\   \ \ , <   /' _ `\   \ \ '__`\  /'__`\     /'_` \
---     \ \ \L\ \\ \ \/     \ \ \\`\ /\ \/\ \   \ \ \L\ \/\ \L\.\_  /\ \L\ \
---      \ \_,__/ \ \_\      \ \_\ \_\ \_\ \_\   \ \_,__/\ \__/.\_\ \ \___,_\
---       \/___/   \/_/       \/_/\/_/\/_/\/_/    \/___/  \/__/\/_/  \/__,_ /

---     _   _
---    | |_| |_  ___
---    |  _| ' \/ -_)
---     \__|_||_\___|

return require"lib".settings[[

brknbad: 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
  -best   -B  best set                       = .5
  -cohen  -c  cohen                          = .35
  -far    -F  how far to go for far          = .9
  -goal   -g  goal                           = recurrence-events
  -K      -K  manage low class counts        = 1
  -leaves -l  number of items in leaves      = .5
  -M      -M  manage low evidence counts     = 2
  -p      -p  coefficient on distance        = 2
  -rule   -r  rule for assessing bins;       = optimize
              one of: (optimize,monitor,tabu)
  -rest   -R  rest is -R*best                = 3
  -some   -s  sample size for distances      = 512
  -seed   -S  seed                           = 10019
  -beam   -T  how many things to call top    = 10
  -wait   -w  wait                           = 10

OPTIONS (other):
  -dump  -d   dump stack on error then quit = false
  -file  -f   file name         = ../etc/data/breastcancer.csv
  -help  -h   show help         = false
  -todo  -t   start up action   = nothing
]]
---      _        _        _               _
---     | |__ _ _| |___ _ | |__  __ _ __| |
---     | '_ \ '_| / / ' \| '_ \/ _` / _` |
---     |_.__/_| |_\_\_||_|_.__/\__,_\__,_|

-- 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 b4={}; for k,_ in pairs(_ENV) do b4[k]=k end
local the, lib, go = require"the", require"lib", require"go"
lib.main(the, go, b4)

---              .---------.
---              |         |
---            -= _________ =-
---               ___   ___
---              |   )=(   |
---               ---   ---
---                  ###
---                #  =  #            "This ain't chemistry.
---                #######             This is art."
---              _
---      __ _| |_____
---     / _` | / / _ \
---     \__,_|_\_\___/

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 or 1 end
ako.xnum   = function(x) return ako.num(x) and not ako.goal(x) end

return ako

---     _                      _  __  _
---    | |___ __ _ _ _ _ _    / |/  \/ |
---    | / -_) _` | '_| ' \   | | () | |
---    |_\___\__,_|_| |_||_|  |_|\__/|_|

local the,_ = require"the", require"lib"
local has2,has3,inc,inc2,inc3   = _.has2,_.has3,_.inc,_.inc2,_.inc3
local push,sort,collect,items   = _.push,_.sort,_.collect,_.items
local map,down1,rnds,oo,class,OBJ = _.map,_.down1,_.rnds,_.oo,_.class,_.OBJ

local NB=class("NB",OBJ)
function NB:new(data, this)
  self.n, self.nh, self.wait         = 0,0, the.wait
  self.e, self.h, self.log,self.cols = {},{},{},nil
  for row in items(data) do
    if   not self.cols
    then self.cols= collect(row,function(j,s) return {name=s,indep=j~=#row} end)
    else self:test(row); self:train(row) end end  end

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

function NB:train(row)
  local more, kl = false, row[#row]
  for col,x in pairs(row) do
    if x ~="?" then
      more = true
      inc3(self.e, col, x, kl)  end end
  if more then
    self.n = self.n + 1
    if not self.h[kl] then self.nh = self.nh + 1 end
    inc(self.h, kl) end end

function NB:classify(t,use)
  local hi,out = -math.huge
  for h,val in pairs(self.h) do
    local prior = ((self.h[h] or 0) + the.K)/(self.n + the.K*self.nh)
    local l = math.log(prior)
    for col,x in pairs(t) do
      if x ~= "?" and self.cols[col].indep then
        l = l + math.log((has3(self.e,col,x,h) + the.M*prior) /
                         ((self.h[h] or 0) + the.M)) end end
    if l>hi then hi,out=l,h end end
  return out end

function NB:score()
  local a,n = 0,#self.log
  for key,x in pairs(self.log) do if x.want==x.got then a=a+1/n end end
  return acc,self.log end

return NB
---     _                     ___  __  _
---    | |___ __ _ _ _ _ _   |_  )/  \/ |
---    | / -_) _` | '_| ' \   / /| () | |
---    |_\___\__,_|_| |_||_| /___|\__/|_|

local R=require
local the,_, ako, NB = R"the",R"lib",R"ako", R"learn101"
local push,items,collect = _.push, _.items, _.collect

return function(data)
  local tmp,xnums = {}
  local function go(c,x,    col)
    if x ~= "?" then
      col = xnums[c]
      if col then x=(x - col.lo) // ((col.hi - col.lo+1E-32) / the.bins) end end
    return x end

  local function xnum(c,name)
    if ako.xnum(name) then return {lo=1E32, hi=-1E32} end end

  local function train(c,x,    col)
    col = xnums[c]
    if col and x ~= "?" then
       col.hi = math.max(x, col.hi)
       col.lo = math.min(x, col.lo) end
    return x end

  print("dat",data)
  for row in items(data) do
    push(tmp, row)
    if   xnums then collect(row, train)
    else xnums = collect(row,xnum)  end end
  for j=2,#tmp do tmp[j] = collect(tmp[j], go) end
  return NB(tmp) end
---     _                     ____ __  _
---    | |___ __ _ _ _ _ _   |__ //  \/ |
---    | / -_) _` | '_| ' \   |_ \ () | |
---    |_\___\__,_|_| |_||_| |___/\__/|_|

local R=require
local nb1,bin,lib  = R"learn101", R"bin", R"lib"
local collect,push = lib.collect,lib.push

return function(data,  log)
  local tmp, xnums = {}
  local function discretize(c,x,   col)
    if x ~= "?" then
      col = xnums[c]
      if col then
        for _,one in pairs(col.bins) do
          if one.lo <= x and x < one.hi then return one.id end end end end
    return x end

  local function xnum(c,name)
    if ako.xnum(name) then return {name=name, xys={},bins={}} end end

  local function train(c,x,row)
    if xnums[c] and x ~= "?" then push(xnums[c].xys, {x=x,y= row[#row]}) end end

  for row in items(data) do
    push(tmp,row)
    if   xnums then collect(row, function(c,x) return train(c,x,row) end)
    else xnums = collect(row,xnum) end end
  for where,col in pairs(xnums) do
    col.bins = bin.Xys(col.xys,where); print(col.name,#col.bins) end
  for j=2,#tmp do tmp[j] = collect(tmp[j], discretize) end
  return nb1(tmp) end
```

```lua
---      _      _
---     | |__  (_)  _ _
---     | '_ \ | | | ' \
---     |_.__/ |_| |_||_|

local _,the,SYM = require"lib", require"the", require"sym"
local fmt,per,upx,push,sort = _.fmt,_.per,_.upx,_.push,_.sort
local ent,o,oo = _.ent,_.o, _.oo
local class,OBJ = _.class, _.OBJ

local BIN=class("BIN",OBJ)
function BIN:new(at,name, lo,hi,ys)
  self.at, self.name         = at or 0, name or ""
  self.lo, self.hi, self.ys = lo, hi or lo, ys or SYM() end

function BIN:__tostring()
  local x,lo,hi,big = self.name, self.lo, self.hi, math.huge
  if     lo ==  hi  then return fmt("%s==%s",x, lo)
  elseif hi ==  big then return fmt("%s>=%s",x, lo)
  elseif lo == -big then return fmt("%s<%s",x, hi)
  else                   return fmt("%s<=%s < %s",lo,x,hi) end end

function BIN:select(row)
  local x, lo, hi = row[self.at], self.lo, self.hi
  return x=="?" or lo == hi and lo == x or lo <= x and x < hi end

function BIN:add(x,y)
  if x<self.lo then self.lo = x end
  if x>self.hi then self.hi = x end
  self.ys:add(y) end


function BIN.mergeSameDivs(b4,after)
  local merged = b4.ys:merged(after.ys)
  if merged then
   return BIN(b4.at, b4.name, b4.lo, after.hi, merged) end end

function BIN.mergeNext(b4,after)
  if b4.hi == after.lo and  b4.lo ~= b4.hi then
   return BIN(b4.at, b4.name, b4.lo, after.hi, b4.ys:merge(after.ys)) end end

return BIN
---                  _
---     _ _  _  _   | |  ___
---    | '_|| || |  | | / -_)
---    |_|   \_,_|  |_| \___|

local R = require
local _,the,COLS,BIN,NUM          = R"lib", R"the", R"cols", R"bin", R"num"
local o,oo,down1,map,push,sort,powerset = _.o,_.oo,_.down1,_.map,_.push,_.sort,_.powerset
local slice,merge,slots,fmt =  _.slice, _.merge,_.slots,_.fmt
local class,OBJ               = _.class, _.OBJ

local RULE = class("RULE",OBJ)

function RULE.best(bins,h)
   local function score1(b1,b2) return RULE({b1},h).score > RULE({b2},h).score end
   return slice(sort(bins, score1), 1, the.beam) end

function RULE.fromBins(bins,h,bests,rests,    n,out,rule,sizes,scores)
  out={}
  sizes=NUM()
  scores=NUM()
  for _,some in pairs(powerset(RULE.best(bins,h))) do
    if #some>0 then
      rule = RULE(some,h)
      sizes:add(#some)
      scores:add(rule.score)
      push(out, {size=#some,score=rule.score,rule=rule}) end end
  local function order(one)
    return ((0 - sizes:norm(one.size))^2 + (1 - scores:norm(one.score))^2)^.5 end
  local n = 0
  for _,three in pairs(sort(out, function(a,b) return order(a) < order(b) end)) do
     local selected1= three.rule:selects(bests)
     local cover1   = 100*#selected1/#bests//1
     local selected2= three.rule:selects(rests)
     local cover2   = 100*#selected2/#rests//1
     if cover1 < 100 or cover2 < 100 then
       print(fmt("%5.3f %4u %4u %s",three.score, cover1, cover2, three.rule))
       n=n+1
       if n > the.beam then return end  end end
  return out end

function RULE:new(bins,h,   t)
  self.seen={}
  self.bins = {}
  for _,bin in pairs(bins) do
    self.bins[bin.at] = self.bins[bin.at] or {}
    push(self.bins[bin.at],  bin) end
  for _,one in pairs(self.bins) do sort(one, function(a,b) return a.lo < b.lo end) end
  self.score = self:scored(h)
  end
function RULE:__tostring() return self:show(self.bins)  end --return self:show(self.bins) end

function RULE:like(klass,h) -- h={"true"=100, "false"=40} n=100+40
  local n=0; for _,v in pairs(h) do n = n + v end
  local fs = {}
  for at,bins in pairs(self.bins) do
    fs[at] = 0
    for _,bin in pairs(bins) do
      fs[at] = fs[at] + (bin.ys.has[klass] or 0) end end
  self.seen[klass] = fs
  local prior = ((h[klass] or 0) + the.K) / (n + the.K * 2)
  local out   = math.log(prior)
  for at,v in pairs(fs) do
     local inc = (v+the.M*prior)/(h[klass]+the.M)
     out=out + math.log( inc)
     end
  return out end

RULE.bias = {}
local bias = RULE.bias
function bias.optimize(b,r) return b+r==0 and 0 or b^2/(b+r) end
function bias.monitor( b,r) return b+r==0 and 0 or r^2/(b+r) end
function bias.tabu(    b,r) return b+r==0 and 0 or 1/(b+r) end

function RULE:scored(h)
  return self.bias[the.rule](self:like("left",h), self:like("right",h)) end

function RULE:selects(rows)
  return map(rows, function(row) if self:select(row) then return row end end) end

function RULE:select(row)
  local function ors(bins)
    for _,bin in pairs(bins) do if bin:select(row) then return true end end
    return false end
  for at,bins in pairs(self.bins) do if not ors(bins) then return false end end
  return true end

function RULE:show(ands)
  local cat, order, sortor, sortand
  cat    = function(t,sep) return table.concat(t,sep) end
  sortand= function(t) return map(slots(t) ,function(k) return t[k] end) end
  sortor = function(a,b)  return a.lo < b.lo end
  return cat(map(sortand(ands),
           function(and1)
            return "("..cat(map(sort(and1,sortor),
                    function(or1) return tostring(or1) end)," or ")..")" end)," and ")
end

-- print has to wipe out fullranges and print selected items
  --sort(bins,order)
  -- ors=   function(bins)
  --          return cat(map(merge(sort(bins,order),BIN.mergeNext))," or ") end
  -- return cat(map(bins, ors)," and ") end

return RULE
```

```lua
---                   _ 
---       __   ___   | |
---      / _| / _ \  | |
---      \__| \___/  |_|

local ako, _ = require"ako", require"lib"
local class, OBJ =  _.class, _.OBJ
local o,oo = _.o, _.oo

local COL = class("COL",OBJ)
function COL:new(at,name)
   self.at, self.name = at or 0, name or ""
   self.n        = 0
   self.ignorep  = ako.ignore(self.name)
   self.indep    = not ako.goal(self.name)
   self.w        = self.name:find"-$" and -1 or 1 end

function COL:adds(t)
  for _,x in pairs(t) do self:add(x) end; return self end

function COL:add(x,inc)
  if x ~= "?" then
    inc = inc or 1
    self.n = self.n + inc
    self:add1(x,inc) end
  return x end

function COL:dist(x,y)
  return x=="?" and y=="?" and 1 or self:dist1(x,y) end

function COL:merged(other,    out)
  out = self:merge(other)
  if out:div()*.95 <= (self.n*self:div() + other.n*other:div())/out.n then
    return out end end

return COL
---       ___  _  _   _ __  
---      (_-< | || | | '  \ 
---      /__/  \_, | |_|_|_|
---            |__/         

local _,ako,COL = require"lib", require"ako", require"COL"
local map,slots,class,ent = _.map,_.slots,_.class,  _.ent

local SYM = class("SYM",COL)
function SYM:new(at,name)
  self:super(at,name)
  self.has, self.most, self.mode = {}, 0, nil end

function SYM:add1(x,inc)
  self.has[x] = inc + (self.has[x] or 0)
  if self.has[x] > self.most then
    self.mode, self.most = x, self.has[x] end end

function SYM:mid()      return self.mode end
function SYM:div()      return ent(self.has, self.n) end
function SYM:same(x,y)  return x==y end
function SYM:dist1(x,y) return self:same(x,y)  and 0 or 1 end

function SYM:like1(x,prior)
  return ((i.has[x] or 0) + the.M*prior)/(self.n + the.M) end

function SYM:merge(other,      out)
  out = SYM(self.at, self.name)
  for x,n in pairs(self.has)  do out:add(x,n) end
  for x,n in pairs(other.has) do out:add(x,n) end
  return out end

function SYM:bins(other, BIN)
  local out = {}
  local function known(x) out[x] = out[x] or BIN(self.at, self.name, x,x) end
  for x,n in pairs(self.has)  do known(x); out[x].ys:add("left", n) end
  for x,n in pairs(other.has) do known(x); out[x].ys:add("right", n) end
  return map(slots(out), function(k) return out[k] end) end

return SYM

---      _ _    _  _   _ __  
---     | ' \  | || | | '  \ 
---     |_||_|  \_,_| |_|_|_|

local _, the, COL = require"lib", require"the", require"col"
local class,merge,per,push,sort,upx = _.class,_.merge,_.per,_.push,_.sort,_.upx
local sd = _.sd
local norm,oo = _.norm,_.oo

local NUM = class("NUM",COL)
function NUM:new(at,name)
  self:super(at,name)
  self.has, self.ok = {}, false
  self.lo, self.hi  = math.huge, -math.huge end

local r=math.random
function NUM:add1(x,inc,        pos)
  for i=1,inc do
    self.lo = math.min(x, self.lo)
    self.hi = math.max(x, self.hi)
    if #self.has < the.some        then pos = 1 + #self.has
    elseif r()   < the.some/self.n then pos = 1 + ((r()*#self.has)//1) end
    if pos then
      self.ok = false
      self.has[pos] = x end end end

function NUM:div(   a) a=self:all(); return (per(a,.9) - per(a,.1))/2.56 end
function NUM:mid()     return per(self:all(), .5)  end
function NUM:same(x,y) return math.abs(x - y) <= the.cohen * self:div() end

function NUM:norm(x) return norm(self.lo, self.hi,x) end

function NUM:dist1(x,y)
  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

function NUM:like1(i,x)
  local sd= self:div()
  if x < self.mu - 4*sd then return 0 end
  if x > self.mu + 4*sd then return 0 end
  local denom = (math.pi*2*sd^2)^.5
  local nom   =  math.exp(1)^(-(x-self.mu)^2/(2*sd^2+1E-32))
  return nom/(denom + 1E-32) end

function NUM:merge(other,   out)
  out = NUM(self.at, self.name)
  for _,x in self(self.has) do out:add(x) end
  for _,x in self(other.has) do out:add(x) end
  return out end

function NUM:all()
  if not self.ok then table.sort(self.has) end
  self.ok=true
  return self.has end

function NUM:bins(other, BIN)
  local tmp,out = {},{}
  for _,x in pairs(self.has ) do push(tmp, {x=x, y="left"}) end
  for _,x in pairs(other.has) do push(tmp, {x=x, y="right"}) end
  tmp = sort(tmp,upx) -- ascending on x
  local now     = push(out, BIN(self.at, self.name, tmp[1].x))
  local epsilon = sd(tmp,function(z) return z.x end) * the.cohen
  local minSize = (#tmp)^the.leaves
  for j,xy in pairs(tmp) do
    if j > minSize and j + minSize < #tmp then -- leave enough for other bins
      if now.ys.n > minSize then                -- enough in this bins
        if xy.x ~= tmp[j+1].x then              -- there is a break in the data
          if now.hi - now.lo > epsilon then    -- "now" not trivially small
            now = push(out,  BIN(self.at, self.name, now.hi)) end end end end
    now:add(xy.x, xy.y) end
  out[1].lo    = -math.huge
  out[#out].hi =  math.huge
  return merge(out, BIN.mergeSameDivs) end

return NUM
```

```lua
---                   _
---      __   ___   | |  ___
---     / _| / _ \  | | (_-<
---     \__| \___/  |_| /__/

local R=require
local _, ako, SYM, NUM  = R"lib", R"ako", R"sym", R"num"
local class, OBJ, push = _.class, _.OBJ, _.push

local COLS = class("COLS",OBJ)
function COLS:new(names)
  self.names, self.klass   = names, nil
  self.all, self.x, self.y = {}, {}, {}
  for at,name in pairs(names) do
    local now = push(self.all, (ako.num(name) and NUM or SYM)(at,name))
    if not ako.ignore(name)  then
      if ako.klass(name) then self.klass=now end
      push(now.indep and self.x or self.y, now) end end end

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

return COLS

```

```lua
---      ___   __ _   ___
---     / -_) / _` | (_-<
---     \___| \__, | /__/
---           |___/

local R = require
local _,the,COLS,BIN                 = R"lib", R"the", R"COLS", R"BIN"
local map,sort,up1,items,push,norm = _.map,_.sort,_.up1,_.items,_.push,_.norm
local items,slice,o,oo,sort,many   = _.items,_.slice,_.o,_.oo,_.sort,_.many
local class,OBJ                    = _.class, _.OBJ

local EGS = class("EGS",OBJ)
function EGS:new() self.rows, self.cols = {}, nil end

function EGS:adds(y) for x in items(y) do self:add(x) end; return self end

function EGS:add(row)
  if not self.cols then self.cols = COLS(row)
                   else push(self.rows, self.cols:add(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:clone(rows)
  local out = EGS()
  out:add(self.cols.names)
  for _,row in pairs(rows or {}) do out:add(row) end
  return out end

function EGS:dist(row1,row2)
  local d, n = 0, 0
  for _,col in pairs(self.cols.x) do
    n = n + 1
    d = d + col:dist(row1[col.at], row2[col.at])^the.p end
  return (d/n) ^ (1/the.p) 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 = norm(col.lo, col.hi, row1[col.at] )
    local b = norm(col.lo, col.hi, 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:bins(other)
  local out = {}
  for n,left in pairs(self.cols.x) do
    local right = other.cols.x[n]
    local tmp   = left:bins(right,BIN)
    if #tmp > 1 then for _,bin in pairs(tmp) do push(out,bin) end end end
  return out end

function EGS:bestRest()
  self.rows = sort(self.rows, function(a,b) return self:better(a,b) end)
  local n = (#self.rows)^the.best
  return slice(self.rows, 1,          n),      -- top n things
              many( self.rows, n*the.rest, n+1) end -- some sample of the rest

-- function egs.xplain(i)
--   best, rest = egs.bestRest(i)
--   return egs.contrasts(i, best,rest) end

return EGS

```

```
---          _              _
---    __  | |  _  _   ___ | |_   ___   _ _
---   / _| | | | || | (_-< |  _| / -_) | '_|
---   \__| |_|  \_,_| /__/  \__| \___| |_|

-- 768
-- | 384
-- | | 192
-- | | | 96
-- | | | | 48           {positive}
-- | | | | 48           {positive}
-- | | | 96
-- | | | | 48           {positive}
-- | | | | 48           {negative}
-- | | 192
-- | | | 96
-- | | | | 48           {positive}
-- | | | | 48           {negative}
-- | | | 96
-- | | | | 48           {positive}
-- | | | | 48           {positive}
-- | 384
-- | | 192
-- | | | 96
-- | | | | 48           {negative}
-- | | | | 48           {negative}
-- | | | 96
-- | | | | 48           {negative}
-- | | | | 48           {negative}
-- | | 192
-- | | | 96
-- | | | | 48           {negative}
-- | | | | 48           {negative}
-- | | | 96
-- | | | | 48           {negative}
-- | | | | 48           {negative}

local R = require
local the,egs,lib = R"the", R"egs", R"lib"
local per,cos,norm,o,fmt,rnds=lib.per,lib.cosine,lib.norm,lib.o,lib.fmt,lib.rnds
local map,any,many,sort,up1 = lib.map,lib.any, lib.many,lib.sort,lib.up1

local cluster={}
function cluster.new(top,egs1,      i,lefts,rights)
  egs1 = egs1 or top
  i   = {egs=egs1, top=top, rank=0}
  lefts, rights, i.left, i.right, i.border, i.c = cluster.half(top, egs1.rows)
  if #egs1.rows >= 2*(#top.rows)^the.leaves then
    if #lefts.rows < #egs1.rows then
      i.lefts = cluster.new(top, lefts)
      i.rights= cluster.new(top, rights) end end
  return i end
---        _  |_    _
---       _>  | |  (_)  \/\/

function cluster.show(i,   pre, front)
  pre = pre or ""
  local front = fmt("%s%s", pre, #i.egs.rows)
  if   cluster.leaf(i)
  then print(fmt("%-20s%s",front, o(rnds(egs.mid(i.egs,i.egs.cols.y)))))
  else print(front)
       if i.lefts  then cluster.show(i.lefts,  "|"..pre)
       if i.rights then cluster.show(i.rights, "|"..pre) end end end end

function cluster.leaf(i) return not (i.lefts or i.rights) end
---        _|  o   _  _|_
---       (_|  |  _>   |_

function cluster.dist(eg1,row1,row2)
  local function sym(c,x,y) return x==y and 0 or 1 end
  local function num(c,x,y)
    if     x=="?" then y = norm(c.lo, c.hi, y); x=y<.5 and 1 or 0
    elseif y=="?" then x = norm(c.lo, c.hi, x); y=x<.5 and 1 or 0
    else              x,y = norm(c.lo, c.hi, x), norm(c.lo, c.hi, y) end
    return math.abs(x-y) end
  local function dist(c,x,y)
    return x=="?" and y=="?" and 1 or (c.nump and num or sym)(c,x,y) end
  local d, n = 0, #eg1.cols.x
  for key,c in pairs(eg1.cols.x) do d=d+dist(c, row1[c.at], row2[c.at])^the.p en
d
  return (d/n)^(1/the.p) end

function cluster.neighbors(eg1, r1, rows)
  return sort(map(rows or eg1.rows,
              function(r2) return {cluster.dist(eg1,r1,r2),r2} end), up1) end
---        _   _   ._   _   ._   _.  _|_  _
---       _>  (/_  |_)  (/_  |   (_|   |_  (/_
---                |

function cluster.half(eg1, rows)
  local project,far,some,left,right,c,lefts,rights,border
  rows    = rows or eg1.rows
  far     = function(r,t) return per(cluster.neighbors(eg1,r,t), the.far)[2] end
  project = function(r)
              return {cos(cluster.dist(eg1,left,r),
                          cluster.dist(eg1,right,r),
                          c),
                      r} end
  some    = many(rows,     the.some)
  left    = far(any(some), some)
  right   = far(left,      some)
  c       = cluster.dist(eg1,left,right)
  lefts,rights = egs.clone(eg1), egs.clone(eg1)
  for n, projection in pairs(sort(map(rows,project), up1)) do
    if n==#rows//2 then border = projection[1] end
    egs.add(n <= #rows//2 and lefts or rights, projection[2]) end
  return lefts, rights, left, right, border, c  end

return cluster

---          _          _
---    __ _ | |__   __  __| |
---   / _` || '_ \ / _|/ _` |
---   \__,_||_.__/ \__|\__,_|

local _,the = require"lib", require"the"
local fmt, inc,slots = _.fmt, _.inc, _.slots
local class,OBJ      = _.class, _.OBJ

local ABCD = class("ABCD",OBJ)

function ABCD:new(data,rx)
  self.data, self.rx = data or "", rx or ""
  self.yes, self.no  = 0,0
  self.known, self.a, self.b, self.c, self.d = {},{},{},{},{} end

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

function ABCD: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,xx in pairs( self.known ) do
    pd,pf,pn,prec,g,f,acc = 0,0,0,0,0,0,0
    a= (self.a[x] or 0); b= (self.b[x] or 0);
    c= (self.c[x] or 0); d= (self.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 self.yes + self.no > 0 then
       acc= self.yes /(self.yes + self.no) end
    out[x] = {data=self.data,rx=self.rx,num=self.yes+self.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 ABCD: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 key,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

function ABCD:adds(gotwants, show)
  for key,one in pairs(gotwants) do
    self:exists(one.want)
    self:exists(one.got)
    if one.want == one.got then self.yes=self.yes+1 else self.no=self.no+1 end
    for x,xx in pairs(self.known) do
      if   one.want == x
      then inc(one.want == one.got and self.d or self.b, x)
      else inc(one.got  == x       and self.c or self.a, x) end end end
  return show and self:pretty(self:report()) or self:report() end

return ABCD
```

```
---      _   _   _
---     | | (_) | |__
---     | | | | | '_ \
---     |_| |_| |_.__/

local lib={}
---      _ _  _ _|_|_  _
---     | | |(_| | | |_\

local r = math.random
function lib.normal(mu,sd)
  mu, sd = (mu or 0), (sd or 1)
  return mu + sd*math.sqrt(-2*math.log(r()))*math.cos(6.2831853*r()) end

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

function lib.norm(lo,hi,x) return math.abs(hi-lo)<1E-9 and 0 or (x-lo)/(hi-lo) end

function lib.ent(t, n)
  if not n then n=0; for _,v in pairs(t) do n=n+v end end
  local e=0;         for _,v in pairs(t) do e=e-v/n*math.log(v/n,2) end
  return e,n end

function lib.sd(sorted, f)
  f=f or function(x) return x end
  local denom = 2.564 -- 2*(1.2 + 0.1*(0.9-0.88493)/(0.9032-0.88493))
  return (f(lib.per(sorted, .9)) - f(lib.per(sorted,.1)))/denom end

function lib.cosine(a,b,c)
  return math.max(0,math.min(1, (a^2+c^2-b^2)/(2*c+1E-32))) end

---      _ |_  _  _ |
---     (_ | |(/_(_ |<

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

---      |`.|-|- _  _. _  _
---     ~|~||  |(/_|  || |(_|
---                         _|

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

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

---     |. __|_ _
---     ||_\ | _\

lib.unpack = table.unpack

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

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

function lib.merge(b4, merge)
  local j,n,tmp = 1,#b4,{}
  while j<=n do
    local a, b = b4[j], b4[j+1]
    if b then
      local c = merge(a, b) -- returns nil if merge fails
      if c then
        a,j = c,j+1 end end
    tmp[#tmp+1] = a
    j = j+1 end
  return #tmp==#b4 and tmp or lib.merge(tmp,merge) end


---      |`.|-|- _  _. _  _
---     ~|~||  |(/_|  || |(_|
---                         _|

function lib.map(t, f, u)
  u={}; for k,v in pairs(t) do u[1+#u]=f(v) end; return u end
function lib.collect(t,f,u)
  u={}; for k,v in pairs(t) do u[k]=f(k,v) end; return u end
function lib.copy(t,   u)
  if type(t) ~= "table" then return t end
  u={}; for k,v in pairs(t) do u[lib.copy(k)] = lib.copy(v) end; return u end

---      _  _  __|_. _  _
---     _\(_)|   |_|| |(_|
---                      _|

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

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

function lib.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 lib.sort(u) end

---       _  -|-  _.  ._  -|-     ._
---      _>   |_ (_|  |    |_  |_| |_)
---                                |

function lib.settings(help)
  local d,used = {},{}
  help:gsub("\n ([-]([^%s]+))[%s]+(-[^%s]+)[^\n]*%s([^%s]+)",
    -- e.g.     "  -bins -b  max.number of bins      = 16"
    --parses to ((-)(bins)) (-b) max number of bins = (16)
    -- i.e.     (long (key)) (short)                  (x)
    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] = lib.coerce(x) end)
  if d.help then os.exit(print(help)) end
  return d end

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

function lib.main(the,go,b4,           resets,todos)
  todos = the.todo == "all" and lib.slots(go) or {the.todo}
  resets={}; for k,v in pairs(the) do resets[k]=v end
  go._fails = 0
  for _,todo in pairs(todos) do
    math.randomseed(the.seed or 10019)
    if go[todo] then print("\n"..todo); go[todo]() end
    for k,v in pairs(resets) do the[k]=v end end
  for k,v in pairs(_ENV) do
    if b4 and not b4[k] then print("?",k,type(v)) end end
  os.exit(go._fails) end

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

function lib.any(a,lo,hi)
  lo,hi = lo or 1, hi or #a; return a[ (lo+(hi-lo)*math.random())//1 ] end

function lib.many(a,n,lo,hi,  u)
  u={}; for j=1,n do lib.push(u, lib.any(a,lo,hi)) end; return u end

function lib.slice(a,lo,hi,    u)
  u, lo, hi = {}, lo or 1, hi or #a
  hi = math.min(hi,#a)
  for j=lo,hi do u[1+#u]=a[j] end; return u end

---      __|_ _. _  _    '~)  -|-|_ . _  _
---     _\ | | || |(_|   /_    | | ||| |(_|
---                  _|                   _|

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

function lib.coerces(s)
  return lib.map(lib.words(s), lib.coerce) end

function lib.coerce(x)
  if type(x) ~= "string" then return x end
  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 lib.items(src,f)
  local function file(f)
    src,f = io.input(src),(f or lib.coerces)
    return function(x) x=io.read()
             if x then return f(x) else io.close(src) end end 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
  if src then
    return type(src) == "string" and file(f) or tbl() end end

---      -|-|_ . _  _  _    '~)   __|_ _. _  _
---       | | ||| |(_|_\    /_   _\ | | || |(_|
---                  _|                       _|

lib.fmt = string.format

function lib.oo(t, slots) print(lib.o(t,slots)) end

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

function lib.dent(t,  seen,pre)
  pre,seen = pre or "", seen or {}
  if seen[t] then t= "..." end
  if type(t)~="table" then return print(pre .. tostring(t)) end
  seen[t]=t
  for key,k in pairs(lib.slots(t)) do
    local v = t[k]
    io.write(lib.fmt("%s:%s%s",pre,k, type(v)=="table" and "\n" or " "))
    if   type(v)=="table"
    then lib.dent(v,seen,"| "..pre)
    else print(v) end end end

function lib.rnds(t,f)
  return lib.map(t, function(x) return lib.rnd(x,f) end) end

function lib.rnd(x,f)
  return lib.fmt(type(x)=="number" and (x~=x//1 and f or "%5.2f") or "%s",x) end

---       _   |_    o   _    _  -|-
---      (_)  |_)   |  (/_  (_   |_
---                _|

local _id=0
function lib.id() _id=_id+1; return _id end

function lib.class(name,base)
  local klass, base_ctor = {}
  if base then
    for k,v in pairs(base) do klass[k] = v end
    klass._base = base
    base_ctor   = rawget(base,'new') end
  klass.__index = klass
  klass._is     = name
  klass._class  = klass
  return setmetatable(klass,{
    __call = function(klass,...)
      local obj = setmetatable({},klass)
      if     rawget(klass,'new')
      then   klass.super = base_ctor
             local res = klass.new(obj,...)
             if res then obj = setmetatable(res,klass) end
      elseif base_ctor then base_ctor(obj,...) end
      return obj end }) end

lib.Obj = lib.class("Obj")

function lib.Obj:show(  t)
  t={}
  for k,v in pairs(self) do if tostring(k):sub(1,1)~="_" then t[1+#t]=k end end
  return lib.sort(t) end

function lib.Obj:__tostring(  u) return lib.o(self,self:show()) end

--u={}; for _,k in pairs(self:show()) do u[1+#u]=lib.fmt(":%s %s",k,self[k]) end
--  return self._is .."{"..table.concat(u," ").."}" end

return lib
```

```lua
---      __ _   ___
---     / _` | / _ \
---     \__, | \___/
---     |___/

local R = require
local _, the, ABCD  = R"lib", R"the", R"ABCD"
local NUM, SYM, BIN,EGS,COLS,RULE = R"num",R"sym",R"bin",R"egs",R"cols",R"rule"
local per,map,dent = _.per, _.map, _.dent

local ish,copy,items,o,oo,powerset = _.ish,_.copy,_.items,_.o,_.oo,_.powerset
local map,fmt,rnds, rnd,push       = _.map,_.fmt,_.rnds, _.rnd,_.push
local class,Obj = _.class, _.Obj
local no,go,ok     = _.no, _.go,_.ok

function go.class()
  local EMP=class("EMP",Obj)
  function EMP:show() return {"name", "age", "_id"} end
  function EMP:new(name) self._id=1;  self.name=name; self.age=0 end
  local fred = EMP("tim")
  local MANAGER=class("MANAGER",EMP)
  local jane = MANAGER("jane")
  print(jane) end

function go.copy(     t,u)
  t={a={b={c=10},d={e=200}}, f=300}
  u= copy(t)
  t.a.b.c= 20
  ok(u.a.b.c ~= 20,"copy") end

function go.rnd()
  ok("23.11" == rnds({23.11111})[1],"rounds") end

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

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

 function go.many( t)
  local o,many=_.o,_.many
  t={};for j = 1,1000 do t[#t+1] = j end
  print(900,"+", o(many(t, 10, 900)))
  print(1,100,   o(many(t, 10,   1, 100)))
  print(300,700, o(many(t, 10, 300, 700))) end

function go.some( n)
  the.some=512
  n=NUM()
  for i=1,999 do n:add( i//100) end
  for k,v in pairs(SYM():adds(n:all()).has) do print(k,v) end end

function go.ent()
  local n = SYM()
  n:add("a",9)
  n:add("b",7)
  ok(ish(n:div(), .98886), "entropy")  end

function go.normal( n)
  n=NUM()
  for i=1,10^3 do n:add( _.normal(10,2) //1) end
  for n,k in pairs(SYM():adds(n:all()).has) do print(n,k) end end

function go.nums( n)
  n=NUM()
  for i=1,10^2 do n:add(_.normal(8,1)) end
  oo(rnds{n:mid(), n:div()}) end

function go.cols()
  _.dent(COLS{"Name","Age:","gender","Weight-"}) end

function go.egs(  i)
  i= EGS():adds(the.file)
  ok(7   == i.cols.x[2].has["lt40"], "counts")
  ok(286 == #i.rows,"egs") end

function go.clone(    i,j)
  i= EGS():adds("../etc/data/auto93.csv")
  j= i:clone(i.rows)
  local flag = true
  for k,n in pairs(i.cols.y[1]:all()) do
    flag=flag and n==j.cols.y[1]:all()[k] end
  ok(flag,"clone") end

function go.mid(     all,best,rest)
  all       = EGS():adds("../etc/data/auto93.csv")
  best,rest = all:bestRest()
  best      = all:clone(best)
  rest      = all:clone(rest)
  print("all ",o(all:mid()))
  print("best",o(best:mid()))
  print("rest",o(rest:mid())) end

function go.bins(     all,best,rest,b4)
  all       = EGS():adds("../etc/data/auto93.csv")
  best,rest = all:bestRest()
  best      = all:clone(best)
  rest      = all:clone(rest)
  for _,bin in pairs(best:bins(rest)) do
    if bin.at ~= b4 then print("") end
    print(bin.name, bin.at,bin.lo,bin.hi,
          bin.ys.has["left"] or 0,
          bin.ys.has["right"] or 0)
    b4 = bin.at end end

local function _rules(file,     all,bests,rests,left,right,b4,bins,rules,h)
  all        = EGS():adds(file)
  bests,rests = all:bestRest()
  left       = all:clone(bests)
  right      = all:clone(rests)
  h          = {left=#bests, right=#rests}
  rules      =  RULE.fromBins(left:bins(right),h,bests,rests)
  end

function go.rules1() _rules("../etc/data/auto93.csv") end
function go.rules2() _rules("../etc/data/china.csv") end
function go.rules3() _rules("../etc/data/nasa93dem.csv") end


local function _dist(file,  i,all)
  local any= _.any
  i= EGS():adds(file)
  local yes=true
  all=NUM()
  for j=1,1000 do
    if (j % 50)==0 then io.write(".") end
    local a,b,c = any(i.rows), any(i.rows), any(i.rows)
    local aa = i:dist(a,a)
    local ba = i:dist(b,a)
    local ab = i:dist(a,b)
    local bc = i:dist(b,c)
    local ac = i:dist(a,c)
    all:adds{aa,ba,ab,bc,ac}
    yes = yes and aa==0 and ab == ba and ab+bc >= ac
    yes = yes and aa>=0 and aa<=1 and ba>=0 and ba<=1 and ab>=0 and ab<=1 and
                  bc>=0 and bc <=1 and ac >= 0 and ac <= 1 end
  oo(rnds(all:all()))
  ok(yes, "dist") end

function go.dist1() _dist(the.file) end
function go.dist2() _dist("../etc/data/diabetes.csv") end

function no.half(  i)
  the.file = "../etc/data/diabetes.csv"
  i = egs.Init(the.file)
  local lefts,rights,left,right,border,c= cluster.half(i)
  print("rows",#i.rows)
  ok(384 == #lefts.rows,  "left")
  ok(384 == #rights.rows, "rights") end

function no.cluster(  i)
  the.file = "../etc/data/diabetes.csv"
  i = egs.Init(the.file)
  cluster.show(cluster.new(i)) end

function go.abcd()
  local t={}
  for _ = 1,6 do push(t,{want="yes",got="yes"}) end
  for _ = 1,2 do push(t,{want="no",got="no"}) end
  for _ = 1,6 do push(t,{want="maybe",got="maybe"}) end
  for _ = 1,1 do push(t,{want="maybe", got="no"}) end
  ABCD():adds(t,true) end

local function qq(i,q)
  print(q[1], fmt("%15s = %-8s best= %s/%s rest= %s/%s",
                  i.cols[q[2]].name, q[3],q[4],q[5],q[6],q[7])) end

local function gonb1(file)
  local i = require"learn101"(file)
  local _, out = i:score()
  local cnt={}
  for _,one in pairs(out) do local k=one.got..",".one.want; cnt[k] = 1+ (cnt[k]
 or 0) end
  for k,n in pairs(cnt) do print(n,o(k)) end
  ABCD():adds(i.log,true) end

function go.nb1a() gonb1(the.file) end
function go.nb1b() gonb1("../etc/data/diabetes.csv") end

function go.nb2()
  the.file = "../etc/data/diabetes.csv"
  the.goal = "positive"
  local i = require("learn201")(the.file);
  ABCD():adds(i.log,true) end

function no.nb2a()
  the.file = "../etc/data/diabetes.csv"
  the.goal = "positive"
  for _,bins in pairs{2,5,9} do
    the.bins = bins
    local i = require("learn201")(the.file);
    ABCD()(i.log,true) end end

function no.nb3()
  the.file = "../etc/data/diabetes.csv"
  the.goal = "positive"
  the.bins = 16
  local i = nb3(the.file);
  abcd(i.log,true)
  local acc, out = score(i);  map(out,function(q) qq(i,q) end) end

return go
```