```lua
---      __                     __
---     /\ \                   /\ \
---     \ \ \____    ____      \ \ \____
---      \ \ '__`\ /'  _ `\     \ \ '__`\
---       \ \ \L\ \/\ \/\ \      \ \ \L\ \
---        \ \_,__/\ \_\ \_\      \ \_,__/
---         \/___/  \/_/\/_/       \/___/

local the,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
  -best  -B   best set                        = .5
  -rest  -R   rest is -R*best                 = 4
  -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 function cli(long,key,short,x)
  local function thing(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 tonumber(x) or x end
  local used={}
  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
   the[key] = thing(x) end

help:gsub("\n ([-]([^%s]+))[%s]+(-[^%s]+)[^\n]*%s([^%s]+)",cli)
if the.help then os.exit(print(help)) end
return the


-- 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 r    = require
local the  = r"the"
local lib  = r"lib"
local abcd = r"abcd"
local bin, rule            = r"bin", r"rule"
local num,sym              = r"num", r"sym"
local ako, egs, summary    = r"ako", r"egs", r"summary"
local learna,learnb,learnc= r"learna", r"learnb", r"learnc"

local ish,items,o,oo,powerset = lib.ish,lib.items,lib.o,lib.oo,lib.powerset
local rnds, rnd = lib.rnds, lib.rnd
-- ## Convenctions:
-- lower case for instance methods, upper case for class methods (e.g.
-- creation, management of sets of instances)
---      _|  _  _ _   _  _
---     (_| (/_| | | (_)_\

local fails=0
local 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

local demo={}
function demo.copy(     t,u)
  t={a={b={c=10},d={e=200}}, f=300}
  u= lib.copy(t)
  t.a.b.c= 20
  print(u.a.b.c)
  oo(t)
  oo(u)
  lib.dent(u)
  end

function demo.rnd()
  oo(rnds{23.1111111}) end

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

function demo.ent()
  local a,b = lib.ent{a=9,b=7}
  print(a,b)
  ok(ish(lib.ent{a=9,b=7}, .98886), "entropy")  end

function demo.items()
  for  x in items{10,20,30} do print(x) end
  local n=0
  print(33)
  for  x in items(the.file) do n=n+1; if n<=5 then print(100); oo(x) end end end

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

 function demo.many( t)
  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 demo.new()
  dent(summary.new{"Name","Age","gender","Weight-"}) end

function demo.clone(   i,t,best,rest, x)
  i={rows={},cols=nil}
  the.file = "../etc/data/auto93.csv"
  bins=xplain(the.file)
  for _,row in pairs(i.rows) do
      x=row[col].at end 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

function demo.nb1()
  local i = nb1(the.file);
  local acc, out = score(i); print(acc); map(out,function(q) qq(i,q) end) end

function demo.nb2()
  the.file = "../etc/data/diabetes.csv"
  the.goal = "positive"
  local i = nb2(the.file);
  abcd(i.log,true) end

function demo.nb2a()
  the.file = "../etc/data/diabetes.csv"
  the.goal = "positive"
  for _,bins in pairs{2,5,9} do
    print(bins)
    the.bins = bins
    local i = nb2(the.file);
    abcd(i.log,true)
    --local acc, out = score(i); print(acc)
    --map(out,function(q) q4(i,q) end)   end end
end end

function demo.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 demo.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

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

fails = 0
local defaults=lib.copy(the)
local todos = defaults.todo == "all" and slots(demo) or {defaults.todo}
for _,todo in pairs(todos) do
  the = lib.copy(defaults)
  math.randomseed(the.seed or 10019)
  print(">>",demo,todo)
  if demo[todo] then demo[todo]() end end

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

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

```lua
---     |  _   _    _  _   _   |       _
---     | (/_ (_| |  |  | (_| . | |_| (_|

local _ = require"lib"
local has2,has3,inc,inc2,inc3,sort = _.has2,_.has3,_.inc,_.inc2,_.inc,_.sort

local function classify(i,t,use)
  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 i.cols[col].indep 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

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

local 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

local 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 r  = has2(i.rest,col,x)
      local r1 = r/i.rests
      local b1 = b/i.bests
      push(out, {100*(b1^2/(b1+r1))//1, col,x,b,i.bests,r,i.rests}) end end
  return acc, sort(out,down1) end

local function nb1(data, log)
  local i = {h={}, nh=0,e={}, n=0, wait=the.wait,
             bests=0,rests=0,best={}, rest={},log=log or {}, cols=nil}
  for row in items(data) do
    if   not i.cols
    then i.cols = collect(row,function(j,s) return {name=s, indep=j~=#row} end)
    else test(i,row); train(i,row) end end
  return i end

return nb1
---     |  _   _    _  _  |_   |       _
---     | (/_ (_| |  |  | |_) . | |_| (_|

local ako = require"ako"
local nb1 = require"learna"

local function nb2(data,  log)
  local tmp,xnums = {}
  local function discretize(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
  -- start
  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], discretize) end
  return nb1(tmp) end

return nb2
---     |  _   _    _  _   _   |       _
---     | (/_ (_| |  |  | (_. | |_| (_|

local nb1  = require"learna"
local lib  = require"lib"
local bin  = require"bin"
local collect,push = lib.collect,lib.push

local function nb3(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

return nb3

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

local bin={}
local the=require"the"
local lib=require"lib"
local fmt,per,push,sort = lib.fmt, lib.per, lib.push, lib.sort

function bin.new(id,at,name,lo,hi,n,div)
  return {id=id,at=at,name=name,lo=lo,hi=hi,n=n,div=div} end

function bin.show(i,negative)
  local x,lo,hi,big, s = i.name, i.lo, i.hi, math.huge
  if negative then
    if     lo== hi  then s=fmt("%s != %s",x,lo)
    elseif hi== big then s=fmt("%s < %s",x,lo)
    elseif lo==-big then s=fmt("%s >= %s",x,hi)
    else                 s=fmt("%s < %s and %s >= %s",x,lo,x,hi) end
  else
    if     lo== hi  then s=fmt("%s == %s",x,lo)
    elseif hi== big then s=fmt("%s >= %s",x,lo)
    elseif lo==-big then s=fmt("%s < %s",x,hi)
    else                 s=fmt("%s <= %s < %s",lo,x,hi) end end
  return s end

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

function bin.Merges(bins)
  local j,n,new = 0,length(bins),{}
  while j <= n do
    j=j+1
    a=bins[j]
    if j < n then
      b = bins[j+1]
      if a.hi == b.lo then
        a.hi  = b.hi
        a.div = (a.div*a.n + b.div*b.n)/(a.n+b.n)
        a.n   = a.n + b.n
        j     = j + 1 end end
    push(new,a) end
  return #new < #bins and bin.Merges(new) or bins end

local _argmin
function bin.Xys(xys,at,name)
  xys                 = sort(xys, upx)
  local triviallySmall = the.cohen*(per(xys,.9).x - per(xys, .1).x)/2.56
  local enoughItems    = #xys / the.bins
  local out            = {}
  _argmin(1,#xys, xys, triviallySmall, enoughItems, -math.huge, at.name, out)
  out[#out].hi =  math.huge
  return out end

function _argmin(lo, hi, xys, triviallySmall, enoughItems, b4, at, name,out)
  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 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*enoughItems then
    for j=lo,hi - enoughItems do
      add(lhs, xys[j].y)
      sub(rhs, xys[j].y)
      local n1,n2 = j - lo +1, hi-j
      if   n1         > enoughItems and
           n2         > enoughItems and
           xys[j].x ~= xys[j+1].x and  -- there is a break here
           xys[j].x  - xys[lo].x > triviallySmall and
           xys[hi].x - xys[j].x  > triviallySmall
      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 -- end if
  if   cut
  then b4 = _argmin(lo,   cut, xys,triviallySmall,enoughItems,b4,at,name,out)
       b4 = _argmin(cut+1,hi , xys,triviallySmall,enoughItems,b4,at,name,out)
  else -- if no cut then the original div was never updates and is still correct
       b4 = push(out,  bin.new(#out+1,at,name,b4,xys[hi].x, hi-lo+1,div)).hi end
  return b4 end

return bin
---       _     |  _   |       _
---     |  |_| | (/_ . | |_| (_|

local rule={}
local lib=require"lib"
local map,push,sort = lib.map, lib.push, lib.sort

function rule.new(bins,   t)
  t = {}
  for _,one in pairs(bins) do t[one.at]=t[one.at] or {}; push(t[one.at],one) end

  return {bins=t} end

function rule.selects(i,row)
  local function ors(bins)
    for _,x in pairs(bins) do if bin.select(x,row) then return true end end
    return false end
  for at,bins in pairs(i.bins) do if not ors(bins) then return false end end
  return true end

function rule.show(i,bins)
  local cat, order, ors
  cat = function(t,sep) return table.concat(t,sep) end
  order= function(a,b)  return a.lo < b.lo end
  ors= function(bins)
           return cat(map(bin.Merges(sort(bins,order)),bin.show)," or ") end
  return cat(map(i.bins, ors)," and ") end

return rule
```

```
---
---     (_)|<(_).||_|(_|

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 num = {}
local ako = require"ako"

function num.new(at,name)
  return {nump=true, indep=false, n=0, at=at or 0, name=name or "",
          w = ako.weight(name or ""), lo=math.huge, hi=-math.huge,
          mu=0,m2=0,sd=0,bins={}} end

function num.add(i,x,   d)
  if x ~= "?" then
    i.n = i.n+1
    i.lo = math.min(x, i.lo)
    i.hi = math.max(x, i.hi)
    d    = x - i.mu
    i.mu = i.mu + d/i.n
    i.m2 = i.m2 + d*(x - i.mu)
    i.sd = ((i.m2<0 or i.n<2) and 0) or ((i.m2/(i.n - 1))^0.5) end
 return x end

return num
---
---     _\\/|_|_|.||_|(_|
---       /
  /
local sym = {}

function sym.new(at,name)
  return {nump=false, indep=false, n=0, at=at or 0,
          name=name or "", has={}, most=0, mode=nil} end

function sym.add(i,x)
  if x ~= "?" then
    i.n = i.n + 1
    i.has[x] = 1 + (i.has[x] or 0)
    if i.has[x] > i.most then
      i.mode,i.most = x,i.has[x] end end
   return x end

return sym
---
---     _\|_||_||_||_|(_||_\/.||_|(_|
---                     /
                  /
local summary = {}
local ako = require"ako"
local sym = require"sym"
local num = require"num"
local lib = require"lib"
local norm= lib.norm

function summary.new(names,    i)
  i = {names={}, klass=nil,xy= {}, x= {}, y={}}
  i.names = names
  for at,name in pairs(names) do
    local now = (ako.num(name) and num.new or sym.new)(at,name)
    push(i.xy, now)
    if not ako.ignore(name)  then
      if not ako.goal(name)  then now.indep = true end
      if     ako.klass(name) then i.klass=now      end
      push(now.indep and i.x or i.y, now)          end end
  return i end

function summary.add(i,row)
  for _,col in pairs(i.xy) do
    (col.nump and num or sym).add(col, row[col.at]) end
  return row end

function summary.better(i,row1,row2)
  local s1, s2, n, e = 0, 0, #i.y, math.exp(1)
  for _,col in pairs(i.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

return summary

---
---     (/_(_|_\.||_|(_|
---          _|
    _|
local egs={}
local summary = require"summary"
local lib     = require"lib"
local map,sort,many = lib.map,lib.sort,lib.many
local items,slice   = lib.items,lib.slice

function egs.new(data,    i)
  i= {rows={}, cols=nil}
  for row in items(data) do
    if not i.cols then i.cols=summary.new(row) else
      push(i.rows, summary.add(i.cols,row)) end end
  return i end

function egs.mid(i,cols)
   local function mid(col) return col.nump and col.mu or col.mode end
   return map(cols or i.cols.y, mid) end

function egs.div(i,cols)
   local function div(col) return col.nump and col.sd or ent(col.has) end
   return map(cols or i.cols.y, div) end

function egs.clone(old,rows)
  local i={rows={}, cols=summary.new(old.cols.names)}
  for _,row in pairs(rows or {}) do summary.add(i.cols,row) end
  return i end

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

function egs.Contrasts(i, rows1, rows2)
  local function contrast(col)
    local function asBin(x,ys,      n,div)
      n,div = ent(ys)
      return bin.new(id, col.at, col.name, x, x, n, div) end
    local symbols, xys, x = {},{}
    for klass,rows in pairs{rows1,rows2} do
      for _,row in pairs(rows) do
        x = row[col.at]
        if x ~= "?" then
          if not col.nump then inc2(symbols,x,klass) end
          push(xys, {x=x, y=klass}) end end end
    return col.nump and bins(xys, col.at) or collect(symbols, asBin) end
  local out, tmp = {}
  for _,col in pairs(i.cols.x) do
    tmp = contrast(col)
    if #tmp > 1 then
      for _,one in pairs(tmp) do push(out, one) end end end
   return out end

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

function egs.dist(i,row1,row2)
  local function sym(_,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, #i.cols.x
  for _,c in pairs(i.cols.x) do d= d + dist(c, row1[c.at], row2[c.at])^the.e end

  return (d/n)^(1/the.e) end

return egs
```

```lua
---    _  |_   _  _| |      _
---   (_| |_) (_ (_| . | |_| (_|

local lib=require"lib"

local 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(lib.slots(t)) do
    local u = t[x]
    print(lib.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

local 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

    -- 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 lib.inc(one.want == one.got and i.d or i.b, x)
      else lib.inc(one.got  == x       and i.c or i.a, x) end end end
  return show and pretty(report()) or report() end

return abcd
```

```lua
---   | * |_  . |      _
---   | | |_) . | |_| (_|

local lib={}

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

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

function lib.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,n end

function lib.norm(lo,hi,x) return math.abs(hi-lo)<1E-9 and 0 or (x-lo)/(hi-lo) 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 aux(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(aux(s), function(a,b) return #a < #b end) 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.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; for j=lo,hi do u[1+#u]=a[j] end; return u end
```

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

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.things(s) return lib.map(lib.words(s), thing) end

function lib.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 lib.items(src,f)
  local function file()
    src,f = io.input(src),f or lib.things
    return function() x=io.read()
             print(6000,f)
             if x then return f(x) else io.close(src) end end end
  local function tbl(   x)
    print(300)
    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() or tbl() end end

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

lib.fmt = string.format

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

function lib.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 lib.o(x, seen) end
  local function show2(k) return lib.fmt(":%s %s",k, lib.o(t[k],seen)) end
  u = #t>0 and lib.map(t,show1) or lib.map(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 _,k in pairs(lib.slots(t)) do
    local v = t[k]
    local after = type(v)=="table" and "\n" or "\t"
    io.write(pre,":",k,after)
    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

return lib
```