```
---     __              __                 __                 __
---    /\ \            /\ \               /\ \               /\ \
---    \ \ \____   _ __\ \ \/'\      ___   \ \ \____     __    \_\ \
---     \ \ '__`\ /\`'__\ \ , <    /' _ `\  \ \ '__`\  /'__`\  /'_` \
---      \ \ \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
  -rest  -R   rest is -R*best            = 4
  -some  -s   sample size for distances  = 512
  -seed  -S   seed                       = 10019
  -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
]]
---     _          _           _               _
---    | |__   _ _| |__ _ _   | |__  __ _   __| |
---    | '_ \ | '_|| / /| ' \  | '_ \/ _` | / _` |
---    |_.__/ |_|  |_\_\|_||_| |_.__/\__,_| \__,_|
```



```
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, lib.go, b4)

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

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

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,new,obj = _.map,_.down1,_.rnds,_.oo,_.new,_.obj

local NB=obj"NB"
function NB:new(data, this)
  this = new(NB,{h={}, nh=0,e={}, n=0, wait=the.wait, log=log or {}, cols=nil})
  for row in items(data) do
    if   not this.cols
    then this.cols= collect(row,function(j,s) return {name=s,indep=j~=#row} end)
    else this:test(row); this:train(row) end end
  return this 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=0
  for key,x in pairs(self.log) do if x.want==x.got then a=a+1/#self.log 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=require"the"
local lib=require"lib"
local fmt,per,upx,push,sort = lib.fmt,lib.per,lib.upx,lib.push,lib.sort
local ent = lib.ent

local bin={}
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
```

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

```lua
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
```

```
---
---      _ _  _ _   _
---     | '_| | ' \ | |  _
---     |_|   \_,_| |_| \___|
```

```lua
local lib=require"lib"
local bin=require"bin"
local map,push,sort = lib.map, lib.push, lib.sort

local rule={}
function rule.new(bins,   t)
  t = {}
  for key,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 key,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
```

```
---
---      _ _
---     | '_ \
---     |_||_|
```

```lua
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
```

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

```lua
local the,ako,_   = require"the", require"ako", require"lib"
local obj,new = _.obj, _.new

local NUM = obj"NUM"
function NUM:new(at,name)
  name=name or ""
  return new(NUM, {at=at or 0, name=name,
                   indep=not ako.goal(name),
                   n=0, has={}, nump=true, n=0, w = ako.weight(name or ""),
                   lo=math.huge, hi=-math.huge, mu=0, m2=0, sd=0, bins={}}) end

function NUM:add(x,   d)
  if x ~= "?" then
    self.n  = self.n+1
    self.lo = math.min(x, self.lo)
    self.hi = math.max(x, self.hi)
    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) end
 return x end

function NUM:div() return i.sd end
function NUM:mid() return i,mu end

function NUM:same(x,y) return math.abs(x - y) <= the.cohen * self.sd end

return NUM
```

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

```lua
local ako,_   =require"ako", require"lib"
local obj,new,ent = _.obj, _.new ,  _.ent

local SYM = obj"SYM"

function SYM:new(at,name)
  name = name or ""
  return new(SYM,{at=at or 0, name=name,
                  nump=false, indep=not ako.goal(name),
                  n=0, has={}, most=0, mode=nil}) 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.mode, self.most = x, self.has[x] end end
   return x end

function SYM:div() return ent(i.has) end
function SYM:mid() return i.mode end

function SYM.merged(i,j,      k)
  k = SYM:new(i.at, i.name)
  for x,n in pairs(i.has) do k:add(x,n) end
  for x,n in pairs(j.has) do k:add(x,n) end
  if ent(k.has) * .99 <= (i.n*ent(i.has) + j.n*ent(j.has))/k.n then
    return k end end

return SYM
```

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

```lua
local R=require
local ako,lib,sym,num = R"ako",R"lib",R"sym",R"num"
local norm,o,oo,push  = lib.norm, lib.o, lib.oo, lib.push

local seen = {}
function seen.new(names)
  return seen.init({names=names, klass=nil,xy= {}, x= {}, y={}},names) end

function seen.init(i, 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 ako.klass(name) then i.klass=now end
      push(now.indep and i.x or i.y, now) end end
  return i end

function seen.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 seen.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 seen
```

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

local R = require
local the,seen,lib     = R"the", R"seen", R"lib"
local map,sort,up1     = lib.map,lib.sort,lib.up1
local items,push,slice = lib.items,lib.push,lib.slice
local o,oo,sort,many   = lib.o,lib.oo,lib.sort,lib.many
---      _ _ _  _ _|_ _
---     (_| (/_(_| | (/_

local egs={}
function egs.new() return {rows={}, cols=nil} end

function egs.Init(data,    i)
  i= egs.new()
  for row in items(data) do
    if  not i.cols then i.cols=seen.new(row) else egs.add(i,row) end end
  return i end

function egs.add(i,row)
  push(i.rows, seen.add(i.cols, row)) 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=seen.new(old.cols.names)}
  for key,row in pairs(rows or {}) do seen.add(i.cols,row) end
  return i end
---      _ _ _ |_ |_ | _ _ |_
---     (_(_)| |_)|_)|_)(_|_\|_|

function egs.bestRest(i)
  i.rows  = sort(i.rows, function(a,b) return seen.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 key,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 key,col in pairs(i.cols.x) do
    tmp = contrast(col)
    if #tmp > 1 then
      for key,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

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 lib=require"lib"
local fmt=lib.fmt

local abcd={}

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

function abcd.exists(i,x,   new)
  new = not i.known[x]
  lib.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 abcd.report(i,    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( 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 abcd.pretty(t)
  print""
  local s1  = "%10s|%10s|%4s|%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(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

function abcd.adds(gotwants, show,data, rx)
  local i = abcd.new(data,rx)
  for key,one in pairs(gotwants) do
    abcd.exists(i,one.want)
    abcd.exists(i,one.got)
    if one.want == one.got then i.yes=i.yes+1 else i.no=i.no+1 end
    for x,xx 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 abcd.pretty(abcd.report(i)) or abcd.report(i) end

return abcd.adds
```

```
---
---      _ _  _ _  _ _
---     | | |(_) |_| | |
---     |_  |_|_|_|_|_|
---       |_|

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) e
nd

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 en
d
function lib.inc2(f,a,b,n)    f=f or{};f[a]=lib.inc(f[a]  or {},b,n); return f en
d
function lib.inc3(f,a,b,c,n) f=f or{};f[a]=lib.inc2(f[a] or{},b,c,n);return f en
d

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.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}
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 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
  if b4 then
    for k,v in pairs(_ENV) do
      if not b4[k] then print("?",k,type(v)) end 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; for j=lo,hi do u[1+#u]=a[j] end; return u end
```

```
828
829 ---     __|_|_|_|_|_|_)   ,~)   _|_|_|_|_|_|_)
830 ---    _\  | | | | | (_|   /_   _\  | | | | | (_|
831 ---                 _|                       _|
832
833 function lib.words(s,sep,    t)
834   sep="([^" .. (sep or ",")  .. "]+)"
835   t={}; for y in s:gmatch(sep) do t[1+#t] = y end; return t end
836
837 function lib.coerces(s)
838   return lib.map(lib.words(s), lib.coerce) end
839
840 function lib.coerce(x)
841   if type(x) ~= "string" then return x end
842   x = x:match"^%s*(.-)%s*$"
843   if x=="true" then return true elseif x=="false" then return false end
844   return math.tointeger(x) or tonumber(x) or x end
845
846 function lib.items(src,f)
847   local function file(f)
848     src,f = io.input(src),(f or lib.coerces)
849     return function(x) x=io.read()
850               if x then return f(x) else io.close(src) end end end
851   local function tbl(    x)
852     x,f = 0, f or function(z) return z end
853     return function() if x< #src then x=x+1; return f(src[x]) end end end
854   if src then
855     return type(src) == "string" and file(f) or tbl() end end
856
857 ---     _|_|_|_|_|_|_)_  _   ,~)   __|_|_|_|_|_|_)
858 ---    _\  | | | | | (_|_\   /_  _\  | | | | | (_|
859 ---                 _|                       _|
860
861 lib.fmt = string.format
862
863 function lib.oo(t) print(lib.o(t)) end
864
865 function lib.o(t,   seen, u)
866   if type(t)~="table" then return tostring(t) end
867   seen = seen or {}
868   if seen[t] then return "..." end
869   seen[t] = t
870   local function show1(x) return lib.o(x, seen) end
871   local function show2(k) return lib.fmt(":%s %s",k, lib.o(t[k],seen)) end
872   u = #t>0 and lib.map(t,show1) or lib.map(lib.slots(t),show2)
873   return (t._is or "")..{"..table.concat(u," ")..} end
874
875 function lib.dent(t,   seen,pre)
876   pre,seen = pre or "", seen or {}
877   if seen[t] then t= "..." end
878   if type(t)~="table" then return print(pre .. tostring(t)) end
879   seen[t]=t
880   for key,k in pairs(lib.slots(t)) do
881     local v = t[k]
882     io.write(lib.fmt("%s:%s%s",pre,k, type(v)=="table" and "\n" or " "))
883     if   type(v)=="table"
884     then lib.dent(v,seen,"| "..pre)
885     else print(v) end end end
886
887 function lib.rnds(t,f)
888   return lib.map(t, function(x) return lib.rnd(x,f) end) end
889
890 function lib.rnd(x,f)
891   return lib.fmt(type(x)=="number" and (x~=x//1 and f or "%5.2f") or "%s",x) end
892
893 ---     _    |_    °    _   _  -|-
894 ---    (_)  |_)   |   (/_  (_  -|-
895 ---               _|
896
897 local _id=0
898 function lib.id() _id=_id+1; return _id end
899
900 function lib.new(x,y) return setmetatable(y,x) end
901
902 function lib.obj(s,   t)
903   t={__tostring=lib.o,_is=s or ""}; t.__index=t
904   return setmetatable(t, {__call=function(...) return t.new(...) end}) end
905
906 return lib
907
```

```
907 ---      __ _   ___
908 ---     / _` | / _ \
909 ---     \__, | \___/
910 ---     |___/
911
912 local R = require
913 local the,_,abcd,bin,rule         = R"the", R"lib",  R"abcd",R"bin",R"rule"
914 local num, sym                    = R"num", R"sym"
915 local ako, egs, seen, cluster     = R"ako", R"egs", R"seen", R"cluster"
916 local learn101, learn201, learn301 = R"learn101", R"learn201", R"learn301"
917
918 local ish,items,o,oo,powerset = _.ish,_.items,_.o,_.oo,_.powerset
919 local map,fmt,rnds, rnd,push       = _.map,_.fmt,_.rnds, _.rnd,_.push
920
921 local go,ok = _.go,_.ok
922
923 function go.copy(     t,u)
924   t={a={b={c=10},d={e=200}}, f=300}
925   u= lib.copy(t)
926   t.a.b.c= 20
927   ok(u.a.b.c ~= 20,"copy") end
928
929 function go.rnd()
930   ok("23.11" == rnds({23.11111})[1],"rounds") end
931
932 function go.collect()
933   local function aux(x,y) return x*y end
934   oo(lib.collect({10,20,30},aux)) end
935
936 function go.ent()
937   local a,b = lib.ent{a=9,b=7}
938   ok(ish(lib.ent{a=9,b=7}, .98886), "entropy")  end
939
940 function go.items()
941   for  x in items{10,20,30} do oo(x) end
942   local n=0
943   for x in items(the.file) do n=n+1; if n<=5 then oo(x) end end end
944
945 function go.powerset()
946   for _,x in pairs(powerset{10,20,30,40,50}) do oo(x) end end
947
948  function go.many( t)
949   local o,many=lib.o,lib.many
950   t={};for j = 1,1000 do t[#t+1] = j end
951   print(900,"+", o(many(t, 10, 900)))
952   print(1,100,    o(many(t, 10,   1, 100)))
953   print(300,700, o(many(t, 10, 300, 700))) end
954
955 function go.new()
956   lib.dent(seen.new{"Name","Age","gender","Weight-"}) end
957
958 -- function go.clone(    i,t,best,rest, x)
959 --   i={rows={},cols=nil}
960 --   the.file = "../etc/data/auto93.csv"
961 --   bins=xplain(the.file)
962 --   for _,row in pairs(i.rows) do
963 --       x=row[col].at end end
964
965 function go.egs(  i)
966   i=egs.Init(the.file)
967   ok(7==i.cols.x[2].has["lt40"], "counts")
968   ok(286 == #i.rows,"egs") end
969
970 function go.dist(  i)
971   local any= lib.any
972   i=egs.Init(the.file)
973   local yes=true
974   for j=1,1000 do
975     if (j % 50)==0 then io.write(".") end
976     local a,b,c = any(i.rows), any(i.rows), any(i.rows)
977     local aa = cluster.dist(i,a,a)
978     local ba = cluster.dist(i,b,a)
979     local ab = cluster.dist(i,a,b)
980     local bc = cluster.dist(i,b,c)
981     local ac = cluster.dist(i,a,c)
982     yes = yes and aa==0 and ab == ba and ab+bc >= ac
983     yes = yes and aa>=0 and aa<=1 and ba>=0 and ba<=1 and ab>=0 and ab<=1 and
984                   bc>=0 and bc <=1 and ac >= 0 and ac <= 1 end
985   ok(yes, "dist") end
986
987 function go.half(  i)
988   the.file = "../etc/data/diabetes.csv"
989   i = egs.Init(the.file)
990   local lefts,rights,left,right,border,c= cluster.half(i)
991   print("rows",#i.rows)
992   ok(384 == #lefts.rows,  "left")
993   ok(384 == #rights.rows, "rights") end
994
995 function go.cluster(  i)
996   the.file = "../etc/data/diabetes.csv"
997   i = egs.Init(the.file)
998   cluster.show(cluster.new(i))
999 end
1000
1001 function go.abcd()
1002   local t={}
1003   for _ = 1,6 do push(t,{want="yes",got="yes"}) end
1004   for _ = 1,2 do push(t,{want="no",got="no"}) end
1005   for _ = 1,6 do push(t,{want="maybe",got="maybe"}) end
1006   for _ = 1,1 do push(t,{want="maybe", got="no"}) end
1007   abcd(t,true) end
1008
1009 local function qq(i,q)
1010   print(q[1], fmt("%15s = %-8s best= %s/%s rest= %s/%s",
1011                   i.cols[q[2]].name, q[3],q[4],q[5],q[6],q[7])) end
1012
1013 local function gonb1(file)
1014   local i = require"learn101"(file)
1015   local _, out = i:score()
1016   local cnt={}
1017   for _,one in pairs(out) do local k=one.got.."."..one.want; cnt[k] = 1+ (cnt[k]
     or 0) end
1018   for k,n in pairs(cnt) do print(n,o(k)) end
1019   abcd(i.log,true) end
1020
1021 function go.nb1a() gonb1(the.file) end
1022 function go.nb1b() gonb1("../etc/data/diabetes.csv") end
1023
1024 function go.nb2()
1025   the.file = "../etc/data/diabetes.csv"
1026   the.goal = "positive"
1027   local i = require("learn201")(the.file);
1028   abcd(i.log,true) end
1029
1030 function go.nb2a()
1031   the.file = "../etc/data/diabetes.csv"
1032   the.goal = "positive"
1033   for _,bins in pairs{2,5,9} do
1034     the.bins = bins
1035     local i = nb2(the.file);
1036     abcd(i.log,true) end end
1037
1038 function go.bins(   t)
1039   local t,n = {},30
1040   for j=1,n do push(t, {x=j, y=j<.6*n and 1 or j<.8*n and 2 or 3}) end
1041   map(bins(t,20),oo) end
```

```lua

function go.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
```