```
-------------------------------------------------------------------------------
---   __                __                  __                    __    --!head

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

---      .-------.
---      | Ba    | Bad <----.  planning= (better - bad)
---      |    56 |          |  monitor = (bad - better)
---      .-------.------.   |
---                | B    |   v
---                |    5 | Better
---                .------.

local b4={}; for k,_ in pairs(_ENV) do b4[k]=k end
local the, help = {}, [[

lua brknbad.lua [OPTIONS]
(c) 2022, Tim Menzies, BSD-2-Clause
Divide things. Show deltas between things.

OPTIONS:
  -cohen     -c cohen                = .35
  -far       -F how far to seek poles = .9
  -keep      -k items to keep        = 256
  -minItems  -m min items in a rang e = .5
  -p         -p euclidean coefficient = 2
  -some      -S sample size for rows  = 512

OPTIONS, other:
  -dump      -d stackdump on error   = false
  -file      -f data file            = ../etc/data/auto93.csv
  -help      -h show help            = false
  -rnd       -r round numbers        = %5.2f
  -seed      -s random number seed   = 10019
  -todo      -t start-up action      = nothing
  -n1        -n1 #repeated trials    = 20
  -n2        -n2 samples per trial   = 100
]]

local any,bestBin,bins,bins1,bootstrap,class,cosine,csv2egs,firsts,fmt,ish
local last,many,map,new,o,ok,oo,optimize,per,pop,push,quintiles,r,rnd,rnds,scott
Knot
local selects,settings,shuffle,slots,smallfx,sort,sum,thing,things,xplains
local NUM,SYM,EGS,BIN,CLUSTER,XPLAIN,GO,NO,OPTIMIZE

--[[                                                                  --!code

## Conventions

### Data

- First row of data are names that describe each column.
- Names ending with `-` or `+` are dependent goals to be minimized or maximized.
- Names ending with `!` are dependent classes.
- Dependent columns are `y` columns (the rest are independent `x` columns).
- Uppercase names are numeric (so the rest are symbolic).
- Names ending with `:'` are columns to be skipped.
- Data is read as rows,  and stored in a EGS instance.
- Within a EGS, row columns are summarized into NUM or SYM instances.

### Inference

- The rows within an EGS are recursive bi-clustered into CLUSTERs
  using random projections (Fastmap) and Aha's distance metric
  (that can process numbers and symbols).
- Entropy-based discretization finds BINs that separates each pair of
  clusters.
- An XPLAIN tree runs the same clustering processing, but data is divided
  at level using the BIN that most separates the clusters.

### Coding

- No globals (so everything is `local`).
- Code 80 characters wide indent with two spaces.
- Format to be read a two-pages-per-page portrait pdf.
- Divide code into section and subsection headings (e.g using figlet)
- Sections are less than 120 lines long (one column in the pdf).
- No lines containing only the word `end` (unless marking the end of a
  complex for loop or function).
- Usually, if an object contains a list of other objects, that sublist
  is called `all`.
- If a slot is too big to display, it is declared private (not to be printed)
  by renaming (e.g.) `slotx` to `_slotx` (so often, `all` becomes `_all`).

### Classes

- Spread class code across different sections (so don't overload reader
  with all details, at one time).
- Show simpler stuff before complex stuff.
- Reserve `i` for `self` (to fit more code per line).
- Don't use inheritance (to simplify readability).
- Use polymorphism (using LUA's  delegation trick).
- Define an class of objects with `Thing=class"thing"` and
  a `function:Thing(args)` creation method.
- Define instances with `new({slot1=value1,slot2=value2,...},Thing)`.
- Instance methods use `.`; e.g. `function Thing.show(i) ... end`.
- Class methods using `:`; e.g.  `Thing:new4strings`. Class methods
  do things like instance creation or manage a set of instances.

### Test suites (and demos)

- Define start-up actions as GO functions.
- In GO functions, check for errors with `ok(test,mdf)`
  (that updates an `fails` counter when not `ok`).
- Define another table called NO so a test can be quickly disabled just
  by renaming it from `GO.xx` to `NO.xx`.

### At top of file

- Trap known globals in `b4`.
- Define all locals at top-of-file (so everyone can access everything).
- Define options in a help string at top of file.
- Define command line options -h (for help); -s (for seeding random numbers)
 `-t` (for startup actions, so `-t all` means "run everything").

### At end of file

- Using `settings`, parse help string to set options,
  maybe updating from command-line.
- Using `GO.main`, run the actions listed on command line.
- `GO.main`  resets random number generator before running an action
- After everything else,  look for `rogues` (any global not in `b4`)
- Finally, return the `fails` as the exit status of this code. --]]
-------------------------------------------------------------------------------
---                                                                   --!misc
---    ___ ___ ___ ___
---    |\/| | [__  |___
---    ___]|___]|___]|___

---    _ _ _ _|_|_|_ _                                                 --!math
---    | | |(_|| | |_\

r=math.random
function ish(x,y,z)    return math.abs(y -x ) < z end
function cosine(a,b,c)
  return math.max(0,math.min(1, (a^2+c^2-b^2)/(2*c+1E-32))) end

---    |_ __|_ _                                                       --!list
---    |_\ |_\

function any(a)        return a[ math.random(#a) ] end
function firsts(a,b)   return a[1] < b[1] end
function last(a)       return a[ #a ] end
function many(a,n,  u) u={}; for j=1,n do push(u,any(a)) end; return u end
function map(t,f, u)   u={};for _,v in pairs(t) do push(u,f(v)) end;return u end
function per(a,p)      return a[ (p*#a)//1 ] end
function pop(a)        return table.remove(a) end
function push(t,x)     t[1 + #t] = x; return x end
function sort(t,f)     table.sort(t,f); return t end
function sum(t,f, n)
  f = f or function(x) return x end
  n=0; for _,v in pairs(t) do n = n + f(v) end; return n end

function shuffle(t,   j)
  for i=#t,2,-1 do j=math.random(i); t[i],t[j]=t[j],t[i] end; return t end

local function quicksort(t,f,lo,hi)
  f= f or function(a,b) return a <= b end
  lo, hi = lo or 1, hi or #t
  if (hi - lo < 1) then return t end
  local j = lo
  for i = lo + 1, hi do
    if f(t[i],t[j]) then
      if   i == j + 1
      then t[j],t[j+1]      = t[j+1],t[j]
      else t[j],t[j+1],t[i] = t[i],t[j],t[j+1] end
    j = j + 1 end
  end
  t = quicksort(t, f, lo, j - 1)
  return quicksort(t, f, j + 1, hi) end
---    __|_ _ _ _|_ _   ,~)   _|_|_ _ _ _ _                            --!s2th
---    _\ | | | |(_|   /_   | | | | |(_|
---                     _|                 _|

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

function things(file,      x)
  local function cells(x,  t)
    t={}; for y in x:gmatch("([^,]+)") do push(t, thing(y)) end; return t end
  file = io.input(file)
  return function()
    x=io.read(); if x then return cells(x) else io.close(file) end end end

---    _|_|_ _ _|_ _   ,~)   _|_|_ _ _ _ _                            --!th2s
---    | | | |(_|   /_  _\  | | | | |(_|
---                     _|                 _|

fmt = string.format

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

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

function slots(t, u)
  u={};for k,v in pairs(t) do if tostring(k):sub(1,1)~="_" then push(u,k)end end
  return sort(u) end

function rnds(t,f) return map(t, function(x) return rnd(x,f) end) end
function rnd(x,f)
  return fmt(type(x)=="number" and (x~=x//1 and f or the.rnd) or "%s",x) end

---    |_|_ _|_|_ _   _|_|_ _ _|_  ,~)   _ _ _|__|_|_|_ _ _ _          --!the
---    | |(/_|| |(_)  (/_><|  /_  _\(/_|__|_|_|(_|_\
---                                                 _|

function settings(help,    d)
  d={}
  help:gsub("\n ([-]([^%s]+))[%s]+(-[^%s]+)[^\n]*%s([^%s]+)",
    function(long,key,short,x)
      for n,flag in ipairs(arg) do
        if flag==short or flag==long then
          x = x=="false" and true or x=="true" and "false" or arg[n+1] end end
      d[key] = x==true and true or thing(x) end)
  if d.help then print(help) end
  return d end

---    _ _ _|_|_ _ _                                                   --!cntr
---    (_(_)| | |(_)|

GO, NO = {fails=0}, {}
function ok(test,msg)
  print(test and "  PASS:"or "  FAIL:",msg or "")
  if not test then
    GO.fails = GO.fails+1
    if the.dump then assert(test,msg) end end end

function GO.main(todo,seed)
  for k,one in pairs(todo=="all" and slots(GO) or {todo}) do
    if k ~= "main" and type(GO[one]) == "function" then
      math.randomseed(seed)
      print(fmt(":%s",one))
      GO[one]() end end
  for k,v in pairs(_ENV) do if not b4[k] then print("?",k,type(v)) end end  end

---    _ |_  _ _ _|_ _                                                 --!obj
---    (_)|_) |(/_(_|_\
---          L|

new = setmetatable
function class(s,   t)
  t={__tostring=o,_is=s or ""}; t.__index=t
  return new(t, {__call=function(_,...) return t.new(_,...) end}) end
```

```lua
-------------------------------------------------------------------------
---
---
---

NUM, SYM, EGS = class"NUM", class"SYM", class"EGS"

---
---

function SYM:new(at,name)
    return new({at=at, name=name, most=0,n=0,all={}}, SYM) end

function NUM:new(at,name)
    return new({at=at, name=name, _all={},
                w=(name or ""):find"-$" and -1 or 1,
                n=0, sd=0, mu=0, m2=0, lo=math.huge, hi=-math.huge}, NUM) end

function EGS:new(names,  i,col)
  i = new({_all={}, cols={names=names, all={}, x={}, y={}}}, EGS)
  for at,name in pairs(names) do
    col = push(i.cols.all, (name:find"^[A-Z]" and NUM or SYM)(at,name) )
    if not name:find":$" then
      if name:find"!$" then i.cols.class = col end
      push(name:find"[-+!]$" and i.cols.y or i.cols.x, col) end end
  return i end

function EGS:new4file(file,  i)
  for row in things(the.file) do
    if i then i:add(row) else i = EGS(row) end end
  return i end

---
---

function SYM.copy(i) return SYM(i.at, i.name) end

function NUM.copy(i) return NUM(i.at, i.name) end

function EGS.copy(i,rows,    j)
  j = EGS(i.cols.names)
  for _,row in pairs(rows or {}) do j:add(row) end
  return j end

---
---
---

function EGS.add(i,row)
  push(i._all,  row)
  for at,col in pairs(i.cols.all) do col:add(row[col.at]) end end

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

function SYM.sub(i,x,inc)
  if x ~= "?" then
    inc = inc or 1
    i.n = i.n - inc
    i.all[x] = i.all[x] - inc end end

function NUM.add(i,x,_,    d,a)
  if x ~="?" then
    i.n   = i.n + 1
    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)
    i.lo  = math.min(x, i.lo)
    i.hi  = math.max(x, i.hi)
    a     = i._all
    if     #a  < the.keep     then i.ok=false; push(a,x)
    elseif r() < the.keep/i.n then i.ok=false; a[r(#a)]=x end end end

function NUM.sub(i,x,_,    d)
  if x ~="?" then
    i.n   = i.n - 1
    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 end

---
---
---

function EGS.mid(i,cols)
  return map(cols or i.cols.y, function(col) return col:mid() end) end

function EGS.div(i,cols)
  return map(cols or i.cols.y, function(col) return col:div() end) end

function NUM.mid(i) return i.mu end
function SYM.mid(i) return i.mode end

function NUM.div(i) return i.sd end
function SYM.div(i,  e)
  e=0; for _,n in pairs(i.all) do
         if n > 0 then e = e - n/i.n * math.log(n/i.n,2) end end
  return math.abs(e) end

function NUM.norm(i,x)
  return i.hi - i.lo < 1E-32 and 0 or (x - i.lo)/(i.hi - i.lo) end

function NUM.all(i)
  if not i.ok then table.sort(i._all); i.ok=true end
  return i._all end

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

--    $ lua brknbad.lua -t cluster
--
--    398
--      | 199
--      | | 99                Weight- Acc+  Mpg+
--      | | | 49              ======= ===== =====
--      | | | | 24            {2542.50 15.68 26.25}
--      | | | | 25            {2408.48 17.72 35.20}
--      | | | 50
--      | | | | 25            {2432.12 16.04 28.80}
--      | | | | 25            {2504.20 16.52 30.80}
--      | | 100
--      | | | 50
--      | | | | 25            {2189.64 16.25 34.00} <== best
--      | | | | 25            {2261.56 16.24 28.80}
--      | | | 50
--      | | | | 25            {2309.24 16.74 26.00}
--      | | | | 25            {2194.60 16.10 26.00}
--      | 199
--      | | 99
--      | | | 49
--      | | | | 24            {3959.83 13.06 14.17}
--      | | | | 25            {4257.64 11.28 12.00} <== worst
--      | | | 50
--      | | | | 25            {3940.24 13.84 19.60}
--      | | | | 25            {4375.32 12.84 13.20}
--      | 100
--      | | | 50
--      | | | | 25            {3220.32 17.40 21.20}
--      | | | | 25            {3259.04 16.39 22.00}
--      | | | 50
--      | | | | 25            {3189.96 16.32 20.00}
--      | | | | 25            {2504.56 16.56 23.20}

CLUSTER=class"CLUSTER"
function CLUSTER:new(top,egs,      i,lefts,rights)
  egs = egs or top
  i   = new({egs=egs, top=top,rank=0},CLUSTER)
  lefts, rights, i.left, i.right, i.border, i.c = top:half(egs._all)
  if #egs._all >= 2*(#top._all)^the.minItems then
    if #lefts._all < #egs._all then
      i.lefts = CLUSTER(top, lefts)
      i.rights= CLUSTER(top, rights) end end
  return i end

function CLUSTER.leaf(i) return not (i.lefts or i.rights) end

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

---
---
---

function EGS.half(i, rows)
  local project,far,some,left,right,c,lefts,rights,border
  rows    = rows or i._all
  far     = function(r,t)  return per(i:dists(r,t), the.far)[2] end
  project = function(r1)
              return {cosine(i:dist(left,r1), i:dist(right,r1), c),r1} end
  some    = many(rows,       the.some)
  left    = far(any(some), some)
  right   = far(left,      some)
  c       = i:dist(left,right)
  lefts,rights = i:copy(), i:copy()
  for n, projection in pairs(sort(map(rows,project),firsts)) do
    if n==#rows//2 then border = projection[1] end
    (n <= #rows//2 and lefts or rights):add( projection[2] ) end
  return lefts, rights, left, right, border, c  end

---
---

function EGS.dists(i,r1,rows)
   return sort(map(rows,function(r2) return {i:dist(r1,r2),r2} end),firsts) end

function EGS.dist(i,row1,row2,    d)
  d = sum(i.cols.x, function(c) return c:dist(row1[c.at], row2[c.at])^the.p end)
  return (d/#i.cols.x)^(1/the.p) end

function NUM.dist(i,a,b)
  if     a=="?" and b=="?" then return 1 end
  if     a=="?" then b=i:norm(b); a=b<.5 and 1 or 0
  elseif b=="?" then a=i:norm(a); b=a<.5 and 1 or 0
  else   a,b = i:norm(a), i:norm(b)  end
  return math.abs(a - b) end

function SYM.dist(i,a,b) return a=="?" and b=="?" and 1 or a==b and 0 or 1 end
```

```
-----------------------------------------------------------------------
---
---     $ lua brknbad.lua -t bins
--
--                          selects  diversity
--                          =======  ========
--           Clndrs   < 5       211   0.48
--           Clndrs  >= 5       187   0.30   <== best overall
--
--            Volume  < 121     158   0.23
--    121  <= Volume  < 168      63   0.84
--    168  <= Volume  < 225      32   0.20
--            Volume >= 225     145   0.00   <== pretty good
--
--            Model   < 73      125   0.87
--    73   <= Model   < 76       91   0.97
--    76   <= Model   < 79       93   1.00
--    Model >= 79                89   0.47
--
--           origin == 1        249   0.72   <== pretty bad
--           origin == 2         70   0.00
--           origin == 3         79   0.00

BIN=class"BIN"
function BIN:new(col,lo,hi,n,div)
  return new({col=col, lo=lo, hi=hi, n=n, div=div},BIN) end

function BIN.selects(i,row,  x)
  x = row[i.col.at]
  return x=="?" or i.lo==i.hi and x==i.lo or i.lo<=x and x<i.hi end

function BIN.show(i,negative)
  local x, lo,hi,big, s = i.col.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.distance2heaven(i, divs, ns)
  return ((1 - ns:norm(i.n))^2 + (0 - divs:norm(i.div))^2)^0.5 end

function BIN:best(bins)
  local divs,ns, distance2heaven = NUM(), NUM()
  function distance2heaven(bin) return {bin:distance2heaven(divs,ns),bin} end
  for _,bin in pairs(bins) do
    divs:add(bin.div); ns:add(  bin.n)
  end
  return sort(map(bins, distance2heaven), firsts)[1][2]  end

function EGS.bins(i,j,  bins)
  bins = {}
  for n,col in pairs(i.cols.x) do
    for _,bin in pairs(col:bins(j.cols.x[n])) do push(bins, bin) end end
  return bins end

---
---
---

function SYM.bins(i,j)
  local xys= {}
  for x,n in pairs(i.all) do push(xys, {x=x,y="left", n=n}) end
  for x,n in pairs(j.all) do push(xys, {x=x,y="right",n=n}) end
  return BIN:new4SYMs(i, SYM, xys) end

function BIN:new4SYMs(col, yclass, xys)
  local out,all={}, {}
  for _,xy in pairs(xys) do
     all[xy.x] = all[xy.x] or yclass()
     all[xy.x]:add(xy.y, xy.n)  end
  for x,one in pairs(all) do push(out,BIN(col, x, x, one.n, one:div())) end
  return out end

---
---

function NUM.bins(i,j)
  local xys, all = {}, NUM()
  for _,n in pairs(i._all) do all:add(n); push(xys,{x=n,y="left"}) end
  for _,n in pairs(j._all) do all:add(n); push(xys,{x=n,y="right"}) end
  return BIN:new4NUMs(i, SYM, sort(xys,function(a,b) return a.x < b.x end),
                      (#xys)^the.minItems, all.sd*the.cohen) end

function BIN:new4NUMs(col, yclass, xys, minItems, cohen)
  local out, b4, argmin = {}, -math.huge
  function argmin(lo,hi)
    local lhs, rhs, cut, div, xpect, xy = yclass(), yclass()
    for j=lo,hi do  rhs:add(xys[j].y) end
    div = rhs:div()
    if hi-lo+1 > 2*minItems
    then
      for j=lo,hi - minItems do
        lhs:add(xys[j].y)
        rhs:sub(xys[j].y)
        if   lhs.n     > minItems and          -- enough items (on left)
             xys[j].x ~= xys[j+1].x and        -- there is a break here
             xys[j].x  - xys[lo].x > cohen and -- not trivially small (on left)
             xys[hi].x - xys[j].x  > cohen     -- not trivially small (on right)
        then xpect = (lhs.n*lhs:div() + rhs.n*rhs:div()) / (lhs.n+rhs.n)
             if xpect < div then               -- cutting here simplifies things
               cut, div = j, xpect end end end --end for
    end -- end if
    if   cut
    then argmin(lo,    cut)
         argmin(cut+1, hi )
    else b4 = push(out, BIN(col, b4, xys[hi].x, hi-lo+1, div)).hi end
  end ----------------------------------------------
  argmin(1,#xys)
  out[#out].hi =  math.huge
  return out end
```

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

--    % lua brknbad.lua -r xplain
--
--                                        Weight- Acc+  Mpg+
--                                        ======= ===== =====
--    398
--     | Clndrs >= 5 : 190
--     | | Model <  73 : 50
--     | | | Volume >= 318 : 29          {4213.93 11.52 12.41}
--     | | | Volume <  318 : 21          {3412.71 14.38 18.10}
--     | | Model >= 73 : 140
--     | | | Model >= 78 : 50            {3354.20 15.68 22.40}
--     | | | | Volume >= 225 : 32        {3554.53 15.69 20.94}
--     | | | Model <  78 : 90
--     | | | | Volume <  262 : 43        {3298.33 16.97 20.00}
--     | | | | | Model >= 75 : 28        {3401.82 17.36 20.00}
--     | | | | Volume >= 262 : 47
--     | | | | | Model <  74 : 20        {4279.05 12.25 12.00} <== worst
--     | | | | | Model >= 74 : 27        {4177.30 13.40 15.93}
--     | Clndrs <  5 : 208
--     | | origin == 3 : 73
--     | | | Model >= 78 : 41            {2176.20 16.37 33.66}
--     | | | | Model >= 80 : 31          {2176.10 16.36 34.84} <=== best
--     | | | Model <  78 : 32            {2155.03 16.41 26.87}
--     | | origin != 3 : 135
--     | | | origin == 2 : 63
--     | | | | Model >= 75 : 36          {2363.81 16.76 30.83}
--     | | | | Model <  75 : 27          {2284.96 16.67 26.30}
--     | | | origin != 2 : 72
--     | | | | Model <  78 : 28          {2319.25 17.11 26.07}
--     | | | | Model >= 78 : 44          {2512.20 16.16 29.77}
--     | | | | | Model >= 80 : 31        {2547.77 16.51 30.00}

XPLAIN=class"XPLAIN"
function XPLAIN:new(top,egs)
  local i,stop,lefts,rights,yes, no
  egs  = egs or top
  i    = new({egs=egs,top=top},XPLAIN)
  stop = (#top._all)^the.minItems
  if #egs._all > 2*stop then
    lefts, rights= top:half(egs._all)
    if #lefts._all < #egs._all then
      i.bin   = BIN:best( lefts:bins(rights) )
      yes, no = top:copy(), top:copy()
      for _,row in pairs(egs._all) do
        (i.bin:selects(row) and yes or no):add(row) end
      if #yes._all > stop then i.yes  = XPLAIN(top, yes) end
      if #no._all  > stop then i.no   = XPLAIN(top, no) end end end
  return i end

function XPLAIN.show(i, pre,how)
  pre, how = pre or "", how or ""
  local front = fmt("%s%s%s", pre, how, #i.egs._all)
  if   i.yes and i.no
  then print(fmt("%-40s",front))
  else print(fmt("%-40s %s",front, o(rnds(i.egs:mid()))))
  end
  if i.yes then i.yes:show("|".. pre, i.bin:show()     ..":") end
  if i.no  then i.no:show( "|".. pre, i.bin:show(true) ..":") end end
```

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

```
OPTIMIZE=class"OPTIMIZE"
function OPTIMIZE:new(top,egs,n)
  local n,egs,i,stop,lefts,rights = n or 1, egs or top
  i = new({rank=n,egs=eg,top=top},OPTIMIZE)
  stop= (#top._all)^the.minItems
  if #egs._all > 2*stop then
     lefts, rights, i.left, i.right, i.border, i.c = top:half(egs._all)
     if   egs:better(i.left,i.right)
     then lefts, rights, i.left, i.right = rights, lefts, i.right, i.left
          i.border = i.c - i.border
     end
     i.rights = OPTIMIZE(top,i.rights,n)  end
     i.rank   = i.rights.n + 1
     i.egs    = i.lefts
  return i end

function EGS.better(i,row1,row2)
  local s1, s2, n, a, b = 0, 0, #i.cols.y
  for _,col in pairs(i.cols.y) do
    a  = col:norm( row1[col.at] )
    b  = col:norm( row2[col.at] )
    s1 = s1 - 2.7183^(col.w * (a - b) / n)
    s2 = s2 - 2.7183^(col.w * (b - a) / n) end
  return  s1 / n < s2 / n  end

function EGS.betters(i,j)
  return i:better(i:mid(i.cols.all), j:mid(j.cols.all)) end

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

function quintiles(ts,width,  nums,out,all,n,m)
  width=width or 32
  nums=NUM(); for _,t in pairs(ts) do
                for _,x in pairs(sort(t)) do add(nums,x) end end
  all,out = nums.all, {}
  for _,t in pairs(ts) do
     local s, where = {}
     where = function(n) return (width*nums:norm(n))//1 end
     for j = 1, width do s[j]=" " end
     for j = where(per(t,.1)), where(per(t,.3)) do s[j]="-" end
     for j = where(per(t,.7)), where(per(t,.9)) do s[j]="-" end
     s[where(per(t, .5))] = "|"
     push(out,{display=table.concat(s),
               data = t,
               pers = map({.1,.3,.5,.7,.9},
                          function(p) return rnd(per(t,p))end)}) end
  return out end

function smallfx(xs,ys,     x,y,lt,gt,n)
  lt,gt,n = 0,0,0
  if #ys > #xs then xs,ys=ys,xs end
  for _,x in pairs(xs) do
    for j=1, math.min(64,#ys) do
      y = any(ys)
      if y<x then lt=lt+1 end
      if y>x then gt=gt+1 end
      n = n+1 end end
  return math.abs(gt - lt) / n <= the.cliffs end

function bootstrap(y0,z0)
  local x, y, z, b4, yhat, zhat, bigger, obs, adds
  function obs(a,b,    c)
    c = math.abs(a.mu - b.mu)
    return (a.sd + b.sd) == 0 and c or c/((x.sd^2/x.n + y.sd^2/y.n)^.5) end
  function adds(t, num)
    num = num or NUM(); map(t, function(x) add(num,x) end); return num end
  y,z    = adds(y0), adds(z0)
  x      = adds(y0, adds(z0))
  b4     = obs(y,z)
  yhat   = map(y._all, function(y1) return y1 - y.mu + x.mu end)
  zhat   = map(z._all, function(z1) return z1 - z.mu + x.mu end)
  bigger = 0
  for j=1,the.boot do
    if obs( adds(many(yhat,#yhat)),  adds(many(zhat,#zhat))) > b4
    then bigger = bigger + 1/the.boot end end
  return bigger >= the.conf end

--- xxx mid has to be per and
-- XXX implement same
-- XXX need tests for stats
function scottKnot(nums,       all,cohen)
  local mid = function (z) return z.some:mid()
  end ------------------------------
  local function summary(i,j,    out)
    out = copy( nums[i] )
    for k = i+1, j do out = out:merge(nums[k]) end
    return out
  end ---------------------------
  local function div(lo,hi,rank,b4,       cut,best,l,l1,r,r1,now)
    best = 0
    for j = lo,hi do
      if j < hi  then
        l   = summary(lo,  j)
        r   = summary(j+1, hi)
        now = (l.n*(mid(l) - mid(b4))^2 + r.n*(mid(r) - mid(b4))^2
              ) / (l.n + r.n)
        if now > best then
          if math.abs(mid(l) - mid(r)) >= cohen then
            cut, best, l1, r1 = j, now, copy(l), copy(r)
    end end end end
    if cut and not l1:same(r1,the) then
      rank = div(lo,    cut, rank, l1) + 1
      rank = div(cut+1, hi,  rank, r1)
    else
      for i = lo,hi do nums[i].rank = rank end end
    return rank
  end ---------------------------------------------------------
  table.sort(nums, function(x,y) return mid(x) < mid(y) end)
  all   = summary(1,#nums)
  cohen = all.sd * the.cohen
  div(1, #nums, 1, all)
  return nums end
```

```
--------------------------------------------------------------------------------
---     ___ ___
---    |_ |_ |__|
---    |__| |__|

function GO.last()
  ok( 30 == last{10,20,30}, "lasts") end

function GO.per(  t)
  t={};for i=1,100 do push(t,i*1000) end
  ok(70000 == per(t,.7), "per") end

function GO.many(  t)
  t={};for i=1,100 do push(t,i) end; many(t,10) end

function GO.sum(  t)
  t={};for i=1,100 do push(t,i) end; ok(5050==sum(t),"sum")end

function GO.shuffle( t, good)
  t={1,2,3,4,5,6,7,8,9}
  good = true
  for j=1,10^5 do
    t= shuffle(t);
    good = good and sum(t)==45,"shuffle "..j end
  ok(good, "shuffling") end

function GO.sample(   m,n)
  m,n = 10^5,NUM(); for i=1,m do n:add(i) end
  for j=.1,.9,.1 do
    print(j,per(n:all(),j),ish(per(n:all(),j),m*j,m*0.05)) end end

function GO.sym(  s)
  s=SYM(); map({1,1,1,1,2,2,3}, function(x) s:add(x) end)
  ok(ish(s:div(),1.378, 0.001), "ent") end

function GO.num( n)
  n=NUM(); map({10, 12, 23, 23, 16, 23, 21, 16}, function(x) n:add(x) end)
  print(n:div())
  ok(ish(n:div(),5.2373, .001), "div") end

function GO.nums( num,t,b4)
  b4,t,num={},{},NUM()
  for j=1,1000 do push(t,100*r()*j) end
  for j=1,#t  do
    num:add(t[j])
    if j%100==0 then    b4[j] =  fmt("%.5f",num:div()) end end
  for j=#t,1,-1 do
    if j%100==0 then ok(b4[j] == fmt("%.5f",num:div()),"div"..j) end
    num:sub(t[j]) end end

function GO.syms( t,b4,s,sym)
  b4,t,sym, s={},{},SYM(), "I have gone to seek a great perhaps."
  t={}; for j=1,20 do s:gsub('.',function(x) t[#t+1]=x end) end
  for j=1,#t  do
    sym:add(t[j])
    if j%100==0 then    b4[j] =  fmt("%.5f",sym:div()) end end
  for j=#t,1,-1 do
    if j%100==0 then ok(b4[j] == fmt("%.5f",sym:div()),"div"..j) end
    sym:sub(t[j]) end
  end

function GO.loader(  num)
  for row in things(the.file) do
    if num then num:add(row[1]) else num=NUM() end end
  ok(ish(num.mu, 5.455,0.001),"loadmu")
  ok(ish(num.sd, 1.701,0.001),"loadsd") end

function GO.egsShow(  e)
  ok(EGS{"name","Age","Weigh-"},"can make EGS?") end

function GO.egsHead( )
  ok(EGS({"name","age","Weight!"}).cols.x,"EGS")  end

function GO.egs(   egs)
  egs = EGS:new4file(the.file)
  ok(ish(egs.cols.x[1].mu, 5.455,0.001),"loadmu")
  ok(ish(egs.cols.x[1].sd, 1.701,0.001),"loadsd") end

function GO.dist(  ds,egs,one,d1,d2,d3,r1,r2,r3)
  egs = EGS:new4file(the.file)
  one = egs._all[1]
  ds={};for j=1,20 do
         push(ds,egs:dist(any(egs._all), any(egs._all))) end
  oo(rnds(sort(ds),"%5.3f"))
  for j=1,10 do
    r1,r2,r3 = any(egs._all), any(egs._all), any(egs._all)
    d1=egs:dist(r1,r2)
    d2=egs:dist(r2,r3)
    d3=egs:dist(r1,r3)
    ok(d1<= 1 and d2 <= 1 and d3 <= 1 and d1>=0 and d2>=0 and d3>=0 and
       egs:dist(r1,r2) == egs:dist(r2,r1) and
       egs:dist(r1,r1) == 0               and
       d3 <= d1+d2,                       "dist"..j)  end end

function GO.half(  egs,lefts,rights)
  egs = EGS:new4file(the.file)
  lefts, rights = egs:half()
  print("before:", o(rnds(egs:mid())))
  print("half1:",  o(rnds( lefts:mid())),
                   egs:betters(lefts,egs) and "better" or "worse")
  print("half2:",  o(rnds(rights:mid())),
                   egs:betters(rights,egs) and "better" or "worse") end

function GO.cluster()
  CLUSTER(EGS:new4file(the.file)):show() end

function GO.bins(    egs,rights,lefts,col2)
  egs= EGS:new4file(the.file)
  lefts, rights = egs:half(egs._all)
  local b4
  for _,bin in pairs(lefts:bins(rights)) do
    if bin.col.name ~= b4 then print"" end
    b4 = bin.col.name
    print(bin:show(), bin.n, rnd(bin.div)) end end

function GO.xplain()
  XPLAIN(EGS:new4file(the.file)):show() end

function GO.optimize(     rows,header)
  rows = {}
  for row in things(the.file) do
    if header then push(rows,row) else header=row end end
  for j=1,the.n1 do
    io.write"."
    rows = shuffle(rows)
    local train = EGS(header)
    local test  = EGS(header)
    for j,row in pairs(rows) do
      (j< #rows/2 and train or test):add(row) end
    CLUSTER(train):leaves()
    local guesses = optimize(train)
    local m,d=0,0
    for i=1,the.n2 do
       local row1= any(test._all)
       local row2= any(test._all)
       if r()> 0.5 ==guesses:better(row1,row2) then
         d = d+1 end
       if test:better(row1,row2)==guesses:better(row1,row2) then
         m =m + 1
       end end
    print(m/the.n2, d/the.n2) end
  end

function GO.cheat(   egs)
        cheat(EGS:new4file(the.file)) end

--------------------------------------------------------------------------------
the = settings(help)
GO.main(the.todo, the.seed)
os.exit(GO.fails)

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