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

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

lua xplan.lua [OPTIONS]
(c) 2022, Tim Menzies, opensource.org/licenses/BSD-2-Clause

OPTIONS:
  -keep      -k items to keep          = 256
  -cohen     -c cohen                  = .35
  -minItems  -m min items in a rang e  = .5
  -p         -p euclidean coefficient  = 3
  -far       -f how far to seek poles  = .9

OPTIONS, other:
  -dump      -d stackdump on error     = false
  -file      -f data file              = ../../etc/data/auto93.csv
  -help      -h show help              = false
]]
local any,bestSpan,bins,bins1,bootstrap,firsts,fmt,last
local many,map,new,o,obj,oo,per,push,quintiles
local selects,slots,smallfx,sort,sum,thing,things,xplains
---    _ _  _  ___  ___     ___ ___  ___  _    ___
---   |\/| |  [__  |         |  |  | |  | |    [__
---   |  | |  ___] |___      |  |__| |__| |___ ___]

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

function last(a)        return a[ #a ] end
function per(a,p)       return a[ (p*#a)//1 ] end
function any(a)         return a[ math.random(#a) ] end
function many(a,n,  u) u={}; for j=1,n do push(u,any(a)) end; return u end

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

function push(t,x)    t[1+#t]=x; return x end
function map(t,f, u) u={};for _,v in pairs(t) do push(u,f(v)) end; return u end
function sum(t,f, n) n={};for _,v in pairs(t) do n = n + f(v) end; return n end

function sort(t,f)    table.sort(t,f); return t end
function firsts(a,b) return a[1] < b[1] end

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

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

function 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

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

fmt = string.format

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

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

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

local go, ok = {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)
       go[one]() end end
   for k,v in pairs(_ENV) do if not b4[k] then print("?",k,type(v)) end end
   return go.fails end

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

new = setmetatable
function obj(   t)
  t={__tostring=o}; t.__index=t
  return new(t, {__call=function(_,...) return t.new(_,...) end}) end
----------------------------------------------------------------------
---     ___ _     ___ ___ ___ ___ ___
---    |    |    |___| [__  [__  |___ [__
---    |___ |___ |   | ___] ___] |___ ___]

local Num, Sym, Egs = obj(), obj(), obj()

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

function Sym:new(at,name)
    return new({at=at, name=name, 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 = (name:find"^[A-Z]" and Num or Sym)(at,name)
    push(i.cols.all, now)
    if not name:find":$" then
      if name:find"!$" then i.cols.class = col
      push(name:find"[+-!]$" and i.cols.y or i.cols.x, col) end 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,all,    j)
  j = Egs(i.cols.name)
  for _,row in pairs(rows or {}) do i:add(row) end
  return j end

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

function Egs.add(i,row)
  i.all[1 + #i.all] = row
  for at,col in pairs(i.cols) 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)
  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)
    if     #i.all < the.keep  then i.ok=false; push(i.all,x)
    elseif r() < the.keep/i.n then i.ok=false; i.all[r(#i.all)]=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 Num.sorted(i)
  if not i.ok then table.sort(i.all); i.ok=true end
  return i.all 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)
  return -sum(i.all,function(n) return n/i.n*math.log(n/i.n,2) end) end

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

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

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

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

function Egs.half(i, rows)
  local project,far,some,left,right,c,lefts,rights
  far    = function(r,t)  return per(i:dists(r,t), the.far)[2] end
  project= function(r1,  a,b)
             a,b = i:dist(left,r1), i:dist(right,r1)
             return {(a^2 + c^2 - b^2)/(2*c), r1} end
  some   = many(rows,       the.some)
  left   = i:far(any(some), some)
  right  = i: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 mid=row end
    (n <= #rows//2 and lefts or rights):add( projection[2] ) end
  return lefts, rights, left, right, mid, c  end

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

function Num.spans(i,j, cuts)
  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 bins(i,cuts,
         bins1(sort(xys,first),(#xys)^the.minItems,all.sd*the.cohen,Sym,{})) end

function bins1(col, old,new)
  if #new>1 then
    new[1].lo    = -math.huge
    new[#new].hi =  math.huge
    for _,cut in pairs(new) do cut.col= col; push(old,cut) end end end

function bins1(xys, minItems, cohen, yclass, cuts, b4)
  local lhs, rhs, b4, cut, div, xpect = yclass(), yclass(), b4 or xys[1].x
  function xpect(i,j) return (i.n*i:div() + j.n*j.div()) / (i.n + j.n) end
  for _,xy in pairs(xys) do rhs:add(xy.y) end
  div = rhs:div()
  for j,xy in pairs(xys) do
    lhs:add(xy.y)
    rhs:sub(xy.y)
    if lhs.n >= minItems and rhs.n >= minItems then
      if xy.x ~= xys[j+1].x then
        if xy.x - xys[1].x >= cohen and xys[#xys].x - xy.x >= cohen then
          if xpect(lhs,rhs) < div then
            cut, div = j, xpect(lhs,rhs) end end end end end
  if   cut
  then local l,r = {},{}
       for n,xy in pairs(xys) do push(n<=cut and l or r, xy) end
       bins1(l, minItems, cohen, yclass, cuts, b4)
       bins1(r, minItems, cohen, yclass, cuts, xys[cut].x)
  else push(cuts, {lo=b4, hi=xys[#xys].x, n=#xys, div=div}) end end

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

local xplain,xplains,selects,spanShow
function Egs.xplain(i,rows)
  local stop,here,left,right,lefts0,rights0,lefts1,rights1
  rows = rows or i.rows
  here = {all=rows}
  stop = (#i.rows)^the.minItems
  if #rows >= 2*stop then
    lefts0, rights0, here.left, here.right, here.mid, here.c  = half(i, rows)
    if #lefts0.rows < #rows then
      cuts = {}
      for j,col in pairs(lefs0.col.x) do col:spans(rights0.col.x[j],cuts) end
      lefts1,rights1 = {},{}
      for _,row in pairs(rows) do
        push(selects(here.selector, row) and lefts1 or rights1, row) end
      if #lefts1  > stop then here.lefts  = xplain(i,lefts1) end
      if #rights1 > stop then here.rights = xplain(i,rights1) end end end
  return here end

function bestSpan(spans)
  local divs,ns,n,div,stats,dist2heaven = Num(), Num()
  function dist2heaven(s) return {((1 - n(s))^2 + (0 - div(s))^2)^.5,s} end
  function div(s)         return divs:norm( s.all:div() ) end
  function n(s)           return   ns:norm( s.all.n     ) end
  for _,s in pairs(spans) do
    add(divs, s.all:div())
    add(ns,   s.all.n) end
  return sort(map(spans, dist2heaven), firsts)[1][2]  end

function selects(span,row,    lo,hi,at,x)
  lo, hi, at = span.lo, span.hi, span.col.at
  x = row[at]
  if x=="?" then return true end
  if lo==hi then return x==lo else return lo <= x and x < hi end end

function xplains(i,format,t,pre,how,    sel,front)
  pre, how = pre or "", how or ""
  if t then
    pre=pre or ""
    front = fmt("%s%s%s %s",pre,how, #t.all, t.c and rnd(t.c) or "")
    if t.lefts and t.rights then print(fmt("%-35s",front)) else
      print(fmt("%-35s %s",front, o(rnds(mids(i,t.all),format))))
    end
    sel = t.selector
    xplains(i,format,t.lefts,  "|".. pre, spanShow(sel).."":")
    xplains(i,format,t.rights, "|".. pre, spanShow(sel,true) ..":") end 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
  local 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
  local 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
```

```
--------------------------------------------------------------------------------
---     ____  ____
---    |  __ |    |
---    |___] |____|

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
    the[key] = x==true and true or thing(x) end)

if the.help then print(help) end
```