```lua
1    -------------------------------------------------------------------------------
2    ---      /_\              /_\
3    ---     / \ \           / \ \                                          ,o88888
4    ---    / \ \ \         / \ \ \                                        ,o8888888'
5    ---   / \__ \ \____  / \ \ \``\                      ,:o:o:oooo.        ,8o88Pd8888"
6    ---   \ \ \L\ \   \/\ \/\ L\ \                  ,.::.::::o:ooooOoOoO. ,oO8O8Pd888'"
7    ---    \ \___\ \    /  \ \___/               ,.:.::o:ooooOoOO8O8OOo.8OOPd8O8O"
8    ---     \/___/  \  /    \/___/             , ..:.::o:ooOoOOOO8OOOOo.FdO8O8"
9    ---                                       , ..:.::o:ooOoOO8O88O8O,COCOO"
10   ---   a little LUA learning library       , . ..:.::o:oooOoO8O8OOCOCO"
11   ---   (c) Tim Menzies 2022, BSD-2         . ..:.::o:ooOoO8O888O8O,COCO"o
12   ---   https://menzies.us/l5               . ..:.::o:ooooOoCoCCC"o:o
13   ---   Share and enjoy                     . ..:.::o:o:,cooooCo"oo:o:
14   ---                                      `  . . ..:.:cocooo"'o:o::'
15   ---                                       .` . ..::cccoc"'o:o:o:::'
16   ---                                    :.:.  ,c:cccc"':.:.:.:.:.'
17   ---                                  ..:.:"'`::::c:"'.:.:.:.:.:.'
18   ---                               ...:.'.:.::::"'  . . . . .'
19   ---                             . . ..:.::.'"'`   .  . . ''
20   ---                             . . . ....'"'
21   ---                             .. . ."'     -hrr-
22   ---                             .
23   ---
24   ---
25   -------------------------------------------------------------------------------
26   local b4={}; for k,_ in pairs(_ENV) do b4[k]=k end
27   local the,help={},[[
28
29   lua l5.lua [OPTIONS]
30   L5 == a very little LUA learning lab
31
32   OPTIONS (inference):                         | DEFAULT
33    -boot   -b P  #bootstrap samples            | 256
34    -cohen  -c F  cohen's small effect size     | .35
35    -cliffs -C F  threshold on Cliff's delta    | .147
36    -far    -F F  look no further than "far"    | .9
37    -keep   -k    items to keep in a number     | 512
38    -leaves -l    leaf size                     | .5
39    -conf   -n F  confidence for stats tests    | .05
40    -p      -p P  distance calcs coefficient    | 2
41    -seed   -S P  random number seed            | 10019
42    -some   -s    look only at "some" items     | 512
43
44   OPTIONS (housekeeping):
45    -dump   -d    on error, exit+ stacktrace    | false
46    -file   -f S  where to get data             | ../etc/data/auto93.csv
47    -help   -h    show help                     | false
48    -rnd    -r S  format string                 | %5.2f
49    -todo   -t S  start-up action               | nothing
50   ]]
51   -------------------------------------------------------------------------------
52   --[[
53   Copyright 2022, Tim Menzies
54
55   Redistribution and use in source and binary forms, with or without
56   modification, are permitted provided that the following conditions
57   are met:
58
59   1. Redistributions of source code must retain the above copyright
60   notice, this list of conditions and the following disclaimer.
61
62   2. Redistributions in binary form must reproduce the above copyright
63   notice, this list of conditions and the following disclaimer in the
64   documentation and/or other materials provided with the distribution.
65
66   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
67   "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
68   LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
69   FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
70   COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
71   INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,
72   BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
73   LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
74   CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
75   LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
76   ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
77   POSSIBILITY OF SUCH DAMAGE.  --]]
78
79   -------------------------------------------------------------------------------
80   -- ## Coding Conventions
81   --
82   -- - All config options in "the" (which is generated by parsing the help text)
83   -- - LOTS OF SHORT FUNCTIONS
84   -- - Line width = 80
85   -- - when you can, write functions down on one line
86   -- - "i" not "self" (so we can fit more on each line)
87   -- - for loop index variables, do not use i. try j,k instead.
88   -- - if something holds a list of thing, name the holding variable "all"
89   -- - no inheritance
90   -- - only define a method if that is for polymorphism
91   -- - all config items into a global "the" variable
92   -- - all the test cases (or demos) are "function Demo.xxx".
93   --   - If test case assertion crashed, add "1" to Demo.fails
94   --   - On exit return the value of Demo.fails as the exit status
95   -- - random seed reset so carefully, just once, at the end of the code.
96   -- - usually, no line with just "end" on it
```

```lua
97   -------------------------------------------------------------------------------
98   ---
99   ---   |_) |‾| |‾|  |‾|
100  ---   |_/ |‾|  |  |‾|
101
102  -- This code reads date from csv files (where "?" denotes "missing value").
103  local is={}
104  function is.missing(x) return x=="?" end
105
106  -- The names on  row1 of that file define the role of that column.
107  -- Names in row1 ending with ":" are to be ignored
108  function is.skip(x) return x:find":$" end
109
110  -- Names in row1 starting in upper case are numbers
111  function is.num(x) return x:find"^[A-Z]" end
112
113  -- Names in row1 ending with "!" are classes.
114  function is.class(x) return x:find"!$" end
115
116  -- Names in row1 ending with "-" are objectives to be minimized.
117  function is.less(x) return x:find"-$" end
118
119  -- Names in row1 ending with "+" are objectives to be maximized.
120  function is.more(x) return x:find"+$" end
121
122  -- Objectives or classes are dependent variables.
123  function is.dependent(x) return is.more(x) or is.less(x) or is.class(x) end
124
125  -- For example, in this data file, we will ignore column 3 (Hp:),
126  -- try to minimize weight (Lbs-) and maximize acceleration and
127  -- miles per hour (Acc+, Mpg+). Also, with one exception (origin),
128  -- everything is numeric. Finally,  there are some missing values on
129  -- lines 3 and lines 7.
130  --
131  --     Clndrs, Weight, Hp:, Lbs-,  Acc+, Model, origin, Mpg+
132  --     8,      304.0,  193, 4732, 18.5, 70,    1,      10
133  --     8,      ?,      215, 4615, 14,   70,    1,      10
134  --     4,      85,     70,  2070, 18.6, 78,    3,      40
135  --     4,      85,     65,  2110, 19.2, 80,    3,      40
136  --     4,      85,     ?,   1835, 17.3, 80,    2,      40
137  --     4,      98,     76,  2144, 14.7, 80,    2,      40
138
139  -------------------------------------------------------------------------------
140  ---
141  ---   |‾| |‾| |‾]  | |‾‾ |‾  ‾| |‾‾
142  ---   |_| |_] |_]  |_|__ |___  | |__]
143
144  local as = setmetatable
145  local function obj(   t)
146    t={__tostring=o}; t.__index=t
147    return as(t, {__call=function(_,...) return t.new(_,...) end}) end
148
149  local Sym = obj() -- Where to summarize symbols
150  function Sym:new(at,s) return as({
151    is="Sym",       -- type
152    at=at or 0,     -- column index
153    name=s or "",   -- column name
154    n=0,            -- number of items summarized in this column
155    all={},         -- all[x] = n means we've seen "n" repeats of "x"
156    most=0,         -- count of the most frequently seen symbol
157    mode=nil        -- the most commonly seen letter
158    }, Sym) end
159
160  local Num = obj() -- Where to summarize numbers
161  function Num:new(at,s) return as({
162    is="Num",       -- type
163    at=at or 0,     -- column index
164    name=s or "",   -- column name
165    n=0,            -- number of items summarizes in this column
166    mu=0,           -- mean (updated incrementally)
167    m2=0,           -- second moment (updated incrementally)
168    sd=0,           -- standard deviation
169    ok=false,       -- true if "all" is sorted
170    all={},         -- a sample of items seen so far
171    lo=1E31,        -- lowest number seen; initially, big so 1st num sends it low
172    hi=-1E31,       -- highest number seen;initially, msall to 2st num sends it hi
173    w=is.less(s or "") and -1 or 1 -- "-1"= minimize and "1"= maximize
174    }, Num) end
175
176  local Egs = obj() -- Where to store examples, summarized into Syms or Nums
177  function Egs:new(names,     i,col,here)  i=as({
178    is="Egs",       -- type
179    all={},         -- all the rows
180    names=names,    -- list of name
181    cols={},        -- list of all columns  (Nums or Syms)
182    x={},           -- independent columns (nothing marked as "skip")
183    y={},           -- dependent columns (nothing marked as "skip")
184    class=nil       -- classes
185    },Egs)
186    for at,name in pairs(names) do
187      col = (is.num(name) and Num or Sym)(at,name)
188      i.cols[1+#i.cols] = col
189      here = is.dependent(name) and i.y or i.x
190      if not is.skip(name) then
191        here[1 + #here] = col
192        if is.class(name) then i.class=col end end end
193    return i end
194  ---
195  ---   ‾|  |‾|‾|‾||‾|‾|
196  ---         |_|
197
198  function Num.clone(i) return Num(i.at, i.name) end
199  function Sym.clone(i) return Sym(i.at, i.name) end
200
201  local data
202  function Egs.clone(i,rows,     copy)
203    copy = Egs(i.names)
204    for _,row in pairs(rows or {}) do  data(copy,row)   end
205    return copy end
```

```lua
local r    = math.random
local fmt = string.format
local unpack = table.unpack
local function push(t,x) table.insert(t,x); return x end
---
--- (✓_) (7_|¯(_(7_

local thing,things,file2things
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(x,sep,   t)
  t={}; for y in x:gmatch(sep or"([^,]+)") do t[1+#t]=thing(y) end
  return t end

function file2things(file,        x)
  file = io.input(file)
  return function()
    x=io.read();
    if x then return things(x) else io.close(file) end end end
---
--- (_| (7_|¯|_ _\(7_|¯|¯
---

local last,per,any,many
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
---
--- |¡_\|¯

local firsts,sort,map,slots,copy
function firsts(a,b)  return a[1] < b[1] end
function sort(t,f)    table.sort(t,f); return t end
function map(t,f,  u)  u={};for k,v in pairs(t) do push(u,f(v)) end; return u end
function slots(t, u,s)
  u={}
  for k,v in pairs(t) do s=tostring(k);if s:sub(1,1)~="_" then push(u,k) end end
  return sort(u) end

function copy(t,    u)
  if type(t)~="table" then return t end
  u={}; for k,v in pairs(t) do u[copy(k)]=copy(v) end
  return setmetatable(u, getmetatable(t)) end

---
--- _)|¯ |¡¯|¯
---

local oo,o, rnd, rnds
function oo(t) print(o(t)) end
function o(t,seen,         key,xseen,u)
  seen = seen or {}
  if type(t)~="table" then return tostring(t) end
  if seen[t]          then return "…" end
  seen[t] = t
  key   = function(k) return fmt(":%s %s",k,o(t[k],seen)) end
  xseen = function(x) return o(x,seen) end
  u = #t>0 and map(t,xseen) or map(slots(t),key)
  return (t.is or "")..'{'..table.concat(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
---
--- _|¯(7__¯|¯  _\|_|¡|¯|¯(7__\

local Demo, ok = {fails=0}
function ok(test,msg)
  print(test and "PASS: "or "FAIL: ",msg or "")
  if not test then
    Demo.fails=Demo.fails+1
    if the.dump then assert(test,msg) end end end

function Demo.main(todo,seed)
  for k,one in pairs(todo=="all" and slots(Demo) or {todo}) do
    if k ~= "main" and type(Demo[one]) == "function" then
      math.randomseed(seed)
      Demo[one]() end end
  for k,v in pairs(_ENV) do if not b4[k] then print("?",k,type(v)) end end
  return Demo.fails end
---
--- _)(_||¯_\(7_  |¯|(7_||_)  _\|¯|¯|¡|¯|(_|
---

local function settings(txt,   d)
  d={}
  txt: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
      if x=="false" then the[key]=false elseif x=="true" then the[key]=true else
        d[key] = tonumber(x) or x end end)
  if d.help then print(txt) end
  return d end
```

```lua
local add
function add(i,x, inc)
  inc = inc or 1
  if not is.missing(x) then
    i.n = i.n + inc
    i:internalAdd(x,inc) end
  return x end

function Sym.internalAdd(i,x,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

function Num.internalAdd(i,x,inc,     d)
  for j=1,inc do
    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.sorted(i)
  if not i.ok then i.all = sort(i.all) end
  i.ok=true
  return i.all end
---
--- |¯|¯|(_|<(7_  (_|(_|¯|¯(_|

local file2Egs -- not "local data" (since defined above)
function data(i,row)
  push(i.all, row)
  for _,col in pairs(i.cols) do add(col, row[col.at]) end
  return i end

function file2Egs(file,   i)
  for row in file2things(file) do
    if i then data(i,row) else i = Egs(row) end end
  return i end
---
--- _\|_||¯|¯||¯¯|(_||¯|7_(7_

local mids
function mids(i,rows,cols) return i:clone(rows):mid(cols) end

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

function Sym.mid(i) return i.mode end
function Num.mid(i) return i.mu end

function Num.div(i) return i.sd end
function Sym.div(i,   e)
  e=0; for _,n in pairs(i.all) do e=e + n/i.n*math.log(n/i.n,2) end
  return -e end
---
--- (_||¡_\|¯(_||¯|(_|(7_

local far,furthest,neighbors,dist
function far(       i,r1,rows,far)
  return per(neighbors(i,r1,rows),far or the.far)[2] end

function furthest( i,r1,rows)
  return last(neighbors(i,r1,rows))[2] end

function neighbors(i,r1,rows)
  return sort(map(rows, function(r2) return {dist(i,r1,r2),r2} end),firsts) end

function dist(i,row1,row2,     d,n,a,b,inc)
  d,n = 0,0
  for _,col in pairs(i.x) do
    a,b = row1[col.at], row2[col.at]
    inc = is.missing(a) and is.missing(b) and 1 or col:dist1(a,b)
    d = d + inc^the.p
    n = n + 1 end
  return (d/n)^(1/the.p) end

function Sym.dist1(i,a,b) return a==b and 0 or 1 end

function Num.dist1(i,a,b)
  if      is.missing(a) then b=i:norm(b); a=b<.5 and 1 or 0
  elseif is.missing(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 Num.norm(i,x)
  return i.hi - i.lo < 1E-32 and 0 or (x - i.lo)/(i.hi - i.lo) end
```

```lua
410  local half, cluster, clusters
411  function half(i, rows,     project,row,some,left,right,lefts,rights,c,mid)
412    function project(row,a,b)
413      a= dist(i,left,row)
414      b= dist(i,right,row)
415      return {(a^2 + c^2 - b^2)/(2*c), row}
416    end -----------------------
417    some  = many(rows,         the.some)
418    left  = furthest(i,any(some),     some)
419    right = furthest(i,left,           some)
420    c     = dist(i,left,right)
421    lefts,rights = {},{}
422    for n, projection in pairs(sort(map(rows,project),firsts)) do
423      if n==#rows//2 then mid=row end
424      push(n <= #rows//2 and lefts or rights, projection[2]) end
425    return lefts, rights, left, right, mid, c   end
426
427  function cluster(i,rows,   here,lefts,rights)
428    rows = rows or i.all
429    here = {all=rows}
430    if #rows >= 2* (#i.all)^the.leaves then
431      lefts, rights, here.left, here.right, here.mid = half(i, rows)
432      if #lefts < #rows then
433        here.lefts  = cluster(i,lefts)
434        here.rights= cluster(i,rights)  end end
435    return here end
436
437  function clusters(i,format,t,pre,    front)
438    if t then
439      pre=pre or ""
440      front = fmt("%s%s",pre,#t.all)
441      if not t.lefts and not t.rights then
442        print(fmt("%-20s%s",front, o(rnds(mids(i,t.all),format))))
443      else
444        print(front)
445        clusters(i,format,t.lefts, "|".. pre)
446        clusters(i,format,t.rights,"|".. pre) end end end
```

```lua
450  local merge,merged,spans,bestSpan
451  function Sym.spans(i, j)
452    local xys,all,one,last,x,y,n = {}, {}
453    for x,n in pairs(i.all) do push(xys, {x,"lefts",n}) end
454    for x,n in pairs(j.all) do push(xys, {x,"rights",n}) end
455    for _,tmp in ipairs(sort(xys,firsts)) do
456      x,y,n = unpack(tmp)
457      if x ~= last then
458        last = x
459        one  = push(all, {lo=x, hi=x, all=Sym(i.at,i.name)}) end
460      add(one.all, y, n) end
461    return all end
462
463  function Num.spans(i, j)
464    local xys,all,lo,hi,gap,one,x,y,n = {}, {}
465    lo,hi = math.min(i.lo, j.lo), math.max(i.hi,j.hi)
466    gap   = (hi - lo) / (6/the.cohen)
467    for _,n in pairs(i.all) do push(xys, {n,"lefts",1}) end
468    for _,n in pairs(j.all) do push(xys, {n,"rights",1}) end
469    one = {lo=lo, hi=lo, all=Sym(i.at,i.name)}
470    all = {one}
471    for _,tmp in ipairs(sort(xys,firsts)) do
472      x,y,n = unpack(tmp)
473      if    one.hi - one.lo > gap
474      then one = push(all, {lo=one.hi, hi=x, all=one.all:clone()}) end
475      one.hi = x
476      add(one.all, y, n) end
477    all       = merge(all)
478    all[1   ].lo = -math.huge
479    all[#all].hi =  math.huge
480    return all end
481
482  function merge(b4,       j,n,now,a,b,both)
483    j, n, now = 0, #b4, {}
484    while j < #b4 do
485      j    = j+1
486      a, b = b4[j], b4[j+1]
487      if   b then
488        both = a.all:merged(b.all)
489        if    both
490        then  a = {lo=a.lo, hi=b.hi, all=both}
491              j = j + 1 end end
492      push(now,a) end
493    return #now == #b4 and b4 or merge(now) end
494
495  -- XXX make .marged and function
496  function Num.merge(i,j,     k)
497    k=i:clone()
498    for _,x in pairs(i.all) do add(k,x) end
499    for _,x in pairs(j.all) do add(k,x) end
500    return k end
501
502  function Sym.merge(i,j,     k)
503    k = i:clone()
504    for x,n in pairs(i.all) do add(k,x,n) end
505    for x,n in pairs(j.all) do add(k,x,n) end
506    return k end
507
508  function Sym.merged(i,j,    k,ei,ej,ek)
509    k = i:merge(j)
510    ei, ej, ek= i:div(), j:div(), k:div()
511    if ek*.99 <= (i.n*ei + j.n*ej)/k.n then return k end end
512
513  function spans(egs1,egs2,        spans,tmp,col1,col2)
514    spans = {}
515    for c,col1 in pairs(egs1.x) do
516      col2 = egs2.x[c]
517      tmp = col1:spans(col2)
518      if #tmp> 1 then
519        for _,one in pairs(tmp) do push(spans,one) end end end
520    return spans end
521
522  function bestSpan(spans,   )
523    local divs,ns,n,div,stats,dist2heaven = Num(), Num()
524    function dist2heaven(s) return {((1 - n(s))^2 + (0 - div(s))^2)^.5,s} end
525    function div(s)          return divs:norm( s.all:div() ) end
526    function n(s)            return   ns:norm( s.all.n      ) end
527    for _,s in pairs(spans) do
528      add(divs, s.all:div())
529      add(ns,   s.all.n) end
530    return sort(map(spans, dist2heaven), firsts)[1][2]   end
```

```lua
535  local xplain,xplains,selects,spanShow
536  function xplain(i,rows,used,
537                  stop,here,left,right,lefts0,rights0,lefts1,rights1)
538    used=used or {}
539    rows = rows or i.all
540    here = {all=rows}
541    stop = (#i.all)^the.leaves
542    if #rows >= 2*stop then
543      lefts0, rights0, here.left, here.right, here.mid, here.c  = half(i, rows)
544      if #lefts0 < #rows then
545        here.selector = bestSpan(spans(i:clone(lefts0),i:clone(rights0)))
546        push(used, {here.selector.all.name, here.selector.lo, here.selector.hi})
547        lefts1,rights1 = {},{}
548        for _,row in pairs(rows) do
549          push(selects(here.selector, row) and lefts1 or rights1, row) end
550        if #lefts1  > stop then here.lefts  = xplain(i,lefts1,used) end
551        if #rights1 > stop then here.rights = xplain(i,rights1,used) end end end
552    return here end
553
554  function xplains(i,format,t,pre,how,    sel,front)
555    pre, how = pre or "", how or ""
556    if t then
557      pre=pre or ""
558      front = fmt("%s%s%s %s",pre,how, #t.all, t.c and rnd(t.c) or "")
559      if t.lefts and t.rights then print(fmt("%-35s",front)) else
560        print(fmt("%-35s %s",front, o(rnds(mids(i,t.all),format))))
561      end
562      sel = t.selector
563      xplains(i,format,t.lefts,  "|".. pre, spanShow(sel)..":")
564      xplains(i,format,t.rights, "|".. pre, spanShow(sel,true) ..":") end end
565
566  function selects(span,row,    lo,hi,at,x)
567    lo, hi, at = span.lo, span.hi, span.all.at
568    x = row[at]
569    if is.missing(x) then return true end
570    if lo==hi then return x==lo else return lo <= x and x < hi end end
571
572  function spanShow(span, negative,    hi,lo,x,big)
573    if not span then return "" end
574    lo, hi, x, big  = span.lo, span.hi, span.all.name, math.huge
575    if    not negative
576    then if lo ==  hi   then return fmt("%s == %s",x,lo)  end
577         if hi ==  big  then return fmt("%s >= %s",x,lo)  end
578         if lo == -big  then return fmt("%s < %s",x,hi)   end
579         return fmt("%s <= %s < %s",lo,x,hi)
580    else if lo ==  hi   then return fmt("%s != %s",x,lo)  end
581         if hi ==  big  then return fmt("%s < %s",x,lo)   end
582         if lo == -big  then return fmt("%s >= %s",x,hi)  end
```

```lua
         return fmt("%s < %s and %s >= %s", x,lo,x,hi)  end end
--- --- .__|_  _|_   __
--- --- _| |_ |_|| |_| _\

local quintiles,smallfx,bootstrap
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
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
    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

---------------------------------------------------------------------------
--- ._ _   _  o ._
--- | | | (_| | | |
---           _|

function Demo.the() oo(the) end

function Demo.many(a)
  a={1,2,3,4,5,6,7,8,9,10}; ok("{10 2 3}" == o(many(a,3)), "manys") end

local function normal(m,s)
  local pi, sqrt, cos, log = math.pi, math.sqrt, math.cos, math.log
  local function z() return sqrt(-2*log(r())) * cos(2* pi * r()) end
  return m + s*z() end

function Demo.tiles()
  local function ns(m,s,r,      u)
    u={}; for j=1,r do u[1+#u] = normal(m,s) end; return u end
  local ts={}
  local m=100
  for mu=8,12,.25 do ts[1+#ts] = ns(mu, 5, m) end
  ts= sort(map(ts,sort), function(a,b) return per(a,.5) < per(b,.5) end)
  for j,one in pairs(quintiles(ts,20)) do
    print(fmt("[%s]",one.display),o(one.pers),
              smallfx( ts[1], ts[j]),
              bootstrap(ts[1], ts[j])) end end

function Demo.stats(  t1,t2,inc,n,a,b)
  for _,n in pairs{20} do --25,50,100,250,500,1000} do
    inc=1
    while inc < 3 do
      print("")
      t1={}; for j=1,n            do push(t1, j*r()) end
      t2={}; for j,x in pairs(t1) do t2[j]=x+inc end
      a,b = smallfx(t1,t2), bootstrap(t1,t2)
      for _,x in pairs(quintiles{t1,t2}) do print(rnd(inc), x.display,a,b) end
      inc = inc*1.1 end end end

function Demo.stats1(x)
  x1={0.34, 0.49 , 0.51 , 0.6}
  x2={  0.6,  0.7 ,  0.8 ,  0.9}
  x3={ 0.15 , 0.25 , 0.4 ,  0.35}
  x4={ 0.6 ,  0.7 ,  0.8  , 0.9}
  x5={0.1 ,  0.2 ,  0.3 ,  0.4}
  print(bootstrap(x5,x3))
  print(bootstrap(x3,x1))
  print(bootstrap(x1,x2))
  print(bootstrap(x2,x4))
end

function Demo.egs()
  ok(5140==file2Egs(the.file).y[1].hi,"reading") end

function Demo.dist(i)
  i = file2Egs(the.file)
  for n,row in pairs(i.all) do print(n,dist(i, i.all[1], row)) end end

function Demo.far(  i,j,row1,row2,row3,d3,d9)
  i = file2Egs(the.file)
  for j=1,10 do
    row1 = any(i.all)
    row2 = far(i,row1, i.all, .9)
    d9   = dist(i,row1,row2)
    row3 = far(i,row1, i.all, .3)
    d3   = dist(i,row1,row3)
    ok(d3 < d9, "closer far") end end

function Demo.half(  i,lefts,rights)
  i = file2Egs(the.file)
  lefts,rights = half(i, i.all)
  oo(mids(i, lefts))
  oo(mids(i, rights))
  end

function Demo.cluster(   i)
  i = file2Egs(the.file)
  clusters(i,"%.0f",cluster(i)) end

function Demo.spans(    i,lefts,rights)
  i = file2Egs(the.file)
  lefts, rights = half(i, i.all)
  oo(bestSpan(spans(i:clone(lefts), i:clone(rights)))) end

function Demo.xplain(    i,j,tmp,lefts,rights,used)
  i = file2Egs(the.file)
  used={}
  xplains(i,"%.0f",xplain(i, i.all,used))
  map(sort(used,function(a,b)
          return ((a[1] < b[1]) or
                  (a[1]==b[1] and a[2] < b[2]) or
                  (a[1]==b[1] and a[2]==b[2] and a[3] < b[3]))end),oo) end


---------------------------------------------------------------------------
the = settings(help)
Demo.main(the.todo, the.seed)
```