```
_=require"tweak"
local NUM=_.NUM
local the,R,rnd,rnds,per     = _.the, _.R,   _.rnd, _.rnds, _.per
local fmt,map,oo,o,push,sort = _.fmt, _.map, _.oo,  _.o,    _.push, _.sort

the.conf=0.05
the.boot=500
the.cliffs=0.147

local function quintiles(ts,width,  nums,out,n,m)
  width=width or 32
  nums=NUM(); for _,t in pairs(ts) do
                for _,x in pairs(sort(t)) do nums:add(x) end end
  out = {}
  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 fmt("%6s",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,         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(); map(t, function(x) num:add(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

-- r ==> rx
function scottKnot(nums,      all,cohen,summary,div)
  function summary(i,j,    out)
    out = copy(nums[i])
    for k = i+1, j do out = out:merge(nums[k]) end
    return out
  end ---------------------------
  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*((l.mu - b4.mu)^2 + r.n*(r.mu - b4.mu))^2)/(l.n+r.n)
        if now > best then
          if math.abs(l.mu - r.mu) >= 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 demo(   normal,stats, mu)
  function normal(mu,sd)
    return mu + sd*math.sqrt(-2*math.log(R()))*math.cos(2*math.pi*R()) end
  stats = {a={},b={},c={},d={},e={}}
  mu=5
  for _,t in pairs(stats) do
    for i=1,100 do t[1+#t]=normal(mu,3) end; mu = mu + 5 end
  for _,x in pairs(quintiles(stats)) do
    print(table.concat(x.pers,"."),x.display) end

```