

```

1 local _, the, COL = require"lib", require"the", require"col"
2 local class,merge,per,push,sort,upx = _.class,_.merge,_.per,_.push,_.sort,_.upx
3 local sd = _.sd
4 local norm,oo = _.norm,_.oo
5
6 local NUM = class("NUM",COL)
7 function NUM:new(at,name)
8   self:super(at,name)
9   self.has, self.ok = {}, false
10  self.lo, self.hi = math.huge, -math.huge end
11
12 local r=math.random
13 function NUM:add1(x,inc,      pos)
14   for i=1,inc do
15     self.lo = math.min(x, self.lo)
16     self.hi = math.max(x, self.hi)
17     if #self.has < the.some then pos = 1 + #self.has
18     elseif r() < the.some/self.n then pos = 1 + ((r() * #self.has) // 1) end
19     if pos then
20       self.ok = false
21       self.has[pos] = x end end end
22
23 function NUM:div( a) a=self:all(); return sd(a) end --(per(a,.9) - per(a,.1))/2.
24 56 end
25 function NUM:mid() return per(self:all(),.5) end
26 function NUM:same(x,y) return math.abs(x - y) <= the.cohen * self:div() end
27
28 function NUM:norm(x) return norm(self.lo, self.hi,x) end
29
30 function NUM:dist1(x,y)
31   if x=="?" then y = self:norm(y); x=y<.5 and 1 or 0
32   elseif y=="?" then x = self:norm(x); y=x<.5 and 1 or 0
33   else x,y = self:norm(x); self:norm(y) end
34   return math.abs(x-y) end
35
36 function NUM:likel(i,x)
37   local sd= self:div()
38   if x < self.mu - 4*sd then return 0 end
39   if x > self.mu + 4*sd then return 0 end
40   local denom = (math.pi*2*sd^2)^.5
41   local nom = math.exp(1)^(-(x-self.mu)^2/(2*sd^2+1E-32))
42   return nom/(denom + 1E-32) end
43
44 function NUM:merge(other, out)
45   out = NUM(self.at, self.name)
46   for _,x in self(self.has) do out:add(x) end
47   for _,x in self(other.has) do out:add(x) end
48   return out end
49
50 function NUM:all()
51   if not self.ok then table.sort(self.has) end
52   self.ok=true
53   return self.has end
54
55 function NUM:bins(other, BIN)
56   local tmp,out = {},{}
57   for _,x in pairs(self.has) do push(tmp, {x=x, y="left"}) end
58   for _,x in pairs(other.has) do push(tmp, {x=x, y="right"}) end
59   tmp = sort(tmp,upx) -- ascending on x
60   local now = push(out, BIN(self.at, self.name, tmp[1].x))
61   local epsilon = sd(tmp,function(z) return z.x end) * the.cohen
62   local minSize = #tmp * the.leaves
63   for j,xy in pairs(tmp) do
64     if j > minSize and j + minSize < #tmp then -- leave enough for other bins
65       if now.ys.n > minSize then -- enough in this bins
66         if xy.x == tmp[j+1].x then -- there is a break in the data
67           if now.hi - now.lo > epsilon then -- "now" not trivially small
68             now = push(out, BIN(self.at, self.name, now.hi)) end end end end
69             now:add(xy.x, xy.y) end
70             out[1].lo = -math.huge
71             out[#out].hi = math.huge
72             return merge(out, BIN.mergeSameDivs) end
73
74 return NUM

```