

```

1  #!/usr/bin/env lua
2  local _=require("lib")
3  local the=_settings{
4
5  L5 : a lean little learning library, in LUA
6  (c) 2022 Tim Menzies <timmeieee.org> BSD-2 license
7
8  USAGE: lua l5.lua [OPTIONS]
9
10 OPTIONS:
11 -e --eg          start-up example          = nothing
12 -b --bins        max number of bins         = 8
13 -d --dump        on test failure, exit with stack dump = false
14 -f --file        file with csv data         = ../data/auto93.csv
15 -F --Far         how far to look for poles (max=1) = .95
16 -h --help        show help                  = false
17 -m --min         min size. If<1 then t*min else min. = 10
18 -n --nums        number of nums to keep      = 512
19 -p --p           distance calculation coefficient = 2
20 -r --rest        size of "rest" set          = 3
21 -s --seed        random number seed         = 10019
22 -S --Sample      how many numbers to keep    = 10000 }}
23
24 local any,cli,copy,csv,lt,many,map= _any,_cli,_copy,_csv,_lt,_many,_map
25 local o,obj,oo,per,push,rnd,roguess= _o,_obj,_oo,_per,_push,_rnd,_roguess
26 local shallowCopy,shuffle,sort= _sort,_shallowCopy,_sort
27 local Data,Num,Row,Some,Sym
28
29 -----
30 Sym=obj"Sym"
31 function Sym:new(c,x) return {at=c or 0,txt=x or "",n=0,has={}} end
32 function Sym:add(x)
33   if x=="?" then self.n =1+self.n;self.has[x]=1+(self.has[x] or 0) end end
34 function Sym:discretize(x) return x end
35 function Sym:dist(v1,v2)
36   return v1=="?" and v2=="?" and 1 or v1==v2 and 0 or 1 end
37 function Sym:entropy(e,e,fun)
38   function fun(p) return p*math.log(p,p,2) end
39   e=0; for _,n in pairs(self.has) do if n>0 then e=e+fun(n/self.n) end end
40   return e end
41
42 -----
43 Some=obj"Some"
44 function Some:new(c,x)
45   return {at=c or 0, txt=x or "",n=0,isSorted=true, _has={}} end
46 function Some:nums()
47   if not self.isSorted then table.sort(self._has) end
48   self.isSorted=true
49   return self._has end
50 function Some:add(v, pos)
51   if v=="?" then
52     self.n=self.n+1
53     if #self._has < the.Sample then pos=1+(#self._has)
54       elseif math.random()<the.Sample/self.n then pos=math.rand(#self._has) end
55     if pos then self.isSorted=false
56       self._has[pos]= v end end end
57
58 -----
59 Num=obj"Num"
60 function Num:new(c,x)
61   return {at=c or 0,txt=x or "",lo=1E32,hi=-1E32, n=0, has=Some(),
62     w=(x or ""):find"$" and -1 or 1} end
63 function Num:add(x)
64   if x=="?" then self.n = self.n+1
65     self.lo = math.min(x,self.lo)
66     self.hi = math.max(x,self.hi)
67     self.has:add(x) end end
68 function Num:norm(n, lo,hi)
69   lo,hi=self.lo,self.hi
70   return n=="?" and n or (hi-lo < 1E-0 and 0 or (n-lo)/(hi-lo + 1E-32)) end
71 function Num:pers(t, a)
72   a=self.has:nums()
73   return map(t,function(p) return per(a,p) end) end
74 function Num:discretize(x, tmp)
75   tmp = (self.hi - self.lo)/(the.bins - 1)
76   return self.lo == self.hi and 1 or math.floor(x/tmp+.5)*tmp end
77
78 function Num:dist(v1,v2)
79   if v1=="?" and v2=="?" then return 1 end
80   v1,v2 = self:norm(v1), self:norm(v2)
81   if v1=="?" then v1 = v2<.5 and 1 or 0 end
82   if v2=="?" then v2 = v1<.5 and 1 or 0 end
83   return math.abs(v1-v2) end
84
85 -----
86 Row=obj"Row"
87 function Row:new(data,t) return {cells=t} end
88 function Row:around(rows)
89   return sort(map(rows, function(r) return {row=r,d=self-r} end),lt"d") end
90 function Row:far(rows)
91   return per(self:around(rows),the.far).row end
92
93 function Row:__sub(row, d,n,d1,n1)
94   d,n = 0,0
95   for i,col in pairs(self._data.cols.x) do
96     di= col:dist(self.cells[col.at], row.cells[col.at])
97     n = n + 1
98     d = d + di^the.p end
99   return (d/n)^(1/the.p) end
100
101 function Row:__lt(row)
102   self.evalued, row.evalued = true,true
103   local s1,s2,d,n,x,y=0,0,0,0
104   local ys = self._data.cols.y
105   for _,col in pairs(ys) do
106     x,y= self.cells[col.at], row.cells[col.at]
107     x,y= col:norm(x), col:norm(y)
108     s1 = s1 - 2.71828*(col.w * (x-y)/#ys)
109     s2 = s2 - 2.71828*(col.w * (y-x)/#ys) end
110   return s1/#ys < s2/#ys end
111
112 function Row:discretize()
113   self.cooked=map(self._data.cols.all,
114     function(col) col:discretize(self.cells[col.at]) end) end
115
116 -----

```

```

117
118 Data=obj"Data"
119 function Data:new(src)
120   self.rows, self.cols = {}, {names=nil,all={},x={},y={}}
121   self:import(src) end
122
123 function Data:import(src)
124   if type(src)=="string"
125     then csv(src, function(row) self:add(row) end)
126   else map(src or {}, function(row) self:add(row) end) end
127   return self end
128
129 function Data:clone( src) return Data({self.cols.names}):import(src) end
130
131 function Data:add(row, id, what)
132   function what(c,x)
133     return {x:find"[A-Z]" and Num or Sym}(c,x) end
134   if #self.cols.all==0
135     then self.cols.names=row
136       for c,x in pairs(row) do
137         local col = push(self.cols.all, what(c,x))
138         if not x:find"[A-Z]" and self.cols.y or self.cols.x, col) end end
139   else row = row.cells and row or Row(self,row)
140     row._data = self
141     push(self.rows, row)
142     for _,col in pairs(self.cols.x, self.cols.y) do
143       for _,col in pairs(cols) do col:add(row.cells[col.at]) end end end end
144
145 function Data:cheat()
146   for i,row in pairs(sort(self.rows)) do
147     row.rank = math.floor(.5+ 100*i/#self.rows)
148     row.evald = false end
149   self.rows = shuffle(self.rows) end
150
151 function Data:half(rows, above, some,x,y,c,rxs,xs,ys)
152   rows = rows or self.rows
153   some = many(rows, the.Sample)
154   x = above or any(some):far(some)
155   y = x:far(some)
156   c = x - y
157   rxs = function(r) return {r=r,x=(r-x)^2 + c^2 - (r-y)^2)/(2*c)} end
158   xs,ys= {},{}
159   for j,rx in pairs(sort(map(rows,rxs),lt"x")) do
160     push(j<=#rows/2 and xs or ys, rx,r) end
161   return {xs=xs, ys=ys, x=x, y=y, c=c} end
162
163 function Data:best(rows, above,stop)
164   rows = rows or self.rows
165   stop = stop or (the.min >=1 and the.min or (#rows)^the.min)
166   if #rows < stop
167     then return rows
168   else local node = self:half(rows,above)
169     if node.x < node.y
170       then return self:best(node.xs, node.x, stop)
171     else return self:best(node.ys, node.y, stop) end end end
172
173 function Data:fours(rows, stop)
174   rows = rows or shallowCopy(self.rows)
175   stop = stop or (the.min >=1 and the.min or (#rows)^the.min)
176   if #rows < stop
177     then return rows
178   else rows = shuffle(rows)
179     fours = {}; for i=1,4 do push(fours, table.remove(rows)) end
180     t = {}
181     for row1 in pairs(rows) do
182       four1 = sort(map(fours,function(row2)
183         t[four1_id] = t[four1_id] or {}
184         push(t[four1_id], four1) end end)
185     self:fours(sort(fours)[1],stop) end
186
187 function Data:discretize()
188   for _,row in pairs(self.rows) do row:discretize() end end
189
190 function Data:xentropy(e,sym)
191   self:discretize()
192   e=0
193   for _,col in pairs(self.cols.x) do
194     sym = Sym()
195     for _,row in pairs(self.rows) do sym:add(row.cooked[col.at]) end
196     e = e + sym:entropy() end
197   return e end
198
199 -----

```

```

200
201 local eg = {}
202 function eg:egs( fails,old)
203   the = cli(the)
204   fails=0
205   old = copy(the)
206   for k,fun in pairs(eg) do
207     if the.eg == "all" or the.eg == k then
208       for k,v in pairs(old) do the[k]=v end
209       math.randomseed(the.seed)
210       print("\n>>>>>",k)
211       if not fun() then fails = fails+1 end end end
212   roguess()
213   os.exit(fails) end
214
215 function eg.the() oo(the); return true end
216
217 function eg.num( z)
218   z=Num(); for i=1,100 do z:add(i) end; print(z); return true end
219
220 function eg.sym( z)
221   z=Sym(); for _,x in pairs{1,1,1,2,2,3} do z:add(x) end;
222   print(z); return true end
223
224 function eg.data( d)
225   d=Data(the.file); map(d.cols.x,print) return true end
226
227 function eg.dist( num,d,r1,r2,r3)
228   d=Data(the.file)
229   num=Num()
230   for i=1,20 do
231     r1=any(d.rows)
232     r2=any(d.rows)
233     r3=r1:far(d.rows)
234     io.write(rnd(r3-r1)," ")
235     num:add(rnd(r2-r1)) end
236   oo(sort(num.has:nums()))
237   print(#d.rows)
238   return true end
239
240 function eg.sort( d)
241   d = Data(the.file)
242   d:cheat()
243   for i=1,#d.rows,32 do print(i,d.rows[i].rank,o(d.rows[i].cells)) end end
244
245 function eg.half( num,tmp)
246   num=Num()
247   for i=1,20 do
248     local d = Data(the.file)
249     d:cheat()
250     tmp=d:best()
251     map(tmp,function(row) num:add(row.rank) end) end
252   print(#tmp,o(num:pers(.1,.3,.5,.7,.9)))
253   return end
254
255 function eg.discretize( d)
256   d=Data(the.file)
257   print(d:xentropy()); return true end
258
259 function eg.fours( d)
260   d=Data(the.file)
261   d:fours() end
262
263 eg:egs()
264
265 -----

```