

```

1 #!/usr/bin/env lua
2 -- vim : ft=lua et sts=2 sw=2 ts=2 :
3 -----
4
5
6
7
8
9
10
11
12
13
14 -- keys0: understand "N" items by peeking at at few (maybe zero) items.
15 -- Copyright 2022, Tim Menzies, MIT license
16
17 -- Permission is hereby granted, free of charge, to any person obtaining a copy
18 -- of this software and associated documentation files (the "Software"), to
19 -- deal in the Software without restriction, including without limitation the
20 -- rights to use, copy, modify, merge, publish, distribute, sublicense, and/or
21 -- sell copies of the Software, and to permit persons to whom the Software is
22 -- furnished to do so, subject to the following conditions:
23
24 -- The above copyright notice and this permission notice shall be included in
25 -- all copies or substantial portions of the Software.
26
27 -- THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
28 -- IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
29 -- FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
30 -- AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
31 -- LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
32 -- FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS
33 -- IN THE SOFTWARE.
34 -----
35
36 local your = {} -- user settings (may be changes from command-line)
37 local our = {} -- system settings (controlled internal to code)
38 our.help = {}
39
40 ./keys0 [OPTIONS]
41 Understand "N" items by peeking at at few (maybe zero) items.
42 (c) 2022, Tim Menzies, opensource.org/licenses/MIT
43
44 -better prune best half of each split : true
45 -Debug one crash, show stackdump : true
46 -dull small effect if 'dull'*sd : .35
47 -far for far, skip after 'far' : .9
48 -file load data from file : ./../data/auto93.csv
49 -h show help : false
50 -goal smile,frown,xplor, doubt : smile
51 -p coefficient on distance calcs : 2
52 -round round numbers to 'round' : 2
53 -seed random number seed : 10019
54 -Some max number items to explore : 512
55 -Tiny bin size = #t^Tiny' : .5
56 -todo start up action : all]]
57
58
59 our.b4={} -- globals known, pre-code. used to find stray globals
60 for k,_ in pairs(_ENV) do our.b4[k]=k end
61
62 local add, any, asserts,coerce, col, copy, csv, defaults, dist
63 local firsts, fmt, klass, map, main, new,o, push, rand, randi, rnd, rnds
64 local same, seconds, slots, sort, xpects
65
66 function klass(s, it)
67   it = {is=s, tostring=o}
68   it.__index = it
69   return setmetatable(it,{__call=function(_,...) return it.new(...) end}) end
70
71 local COLS,EG,EGS = klass"COLS", klass"EG", klass"EGS"
72 local NUM,RANGE,SYM = klass"NUM", klass"RANGE", klass"SYM"
73
74
75 -----
76
77 function NUM.new(at,s, i)
78   i= new(NUM,{n=0,at=at or 0, txt=s or "",mu=0,m2=0,lo=math.huge,hi=-math.huge})
79   i.w = i.txt:find"-" and -1 or 1
80   return i end
81
82 function NUM.add(i,x, d)
83   if x=="?" then
84     i.n = i.n + 1
85     d = x - i.mu
86     i.mu = i.mu + d/i.n
87     i.m2 = i.m2 + d*(x-i.mu)
88     i.lo = math.min(i.lo,x); i.hi = math.max(i.hi,x) end
89   return x end
90
91 function NUM.dist(i,x,y)
92   if x=="?" and y=="?" then return 1
93   elseif x=="?" then y= i:norm(y); x=y>.5 and 0 or 1
94   elseif y=="?" then x= i:norm(x); y=x>.5 and 0 or 1
95   else x,y = i:norm(x), i:norm(y) end
96   return math.abs(x-y) end
97
98 function NUM.div(i) return i.n<2 and 0 or (i.m2/(i.n-1))^0.5 end
99
100 function NUM.mid(i) return i.mu end
101
102 function NUM.norm(i,x) return i.hi-i.lo<1E-9 and 0 or (x-i.lo)/(i.hi-i.lo) end
103
104 function NUM.ranges(i,j,is,ys)
105   local xys,ranges,merge,div
106   function merge(b4, j,tmp,now,after,maybe)
107     j, tmp = 0, {}
108     while j < #b4 do
109       j = j + 1
110       now, after = b4[j], b4[j+1]
111       if after then
112         maybe = now:merge(after)
113         if maybe then now=maybe; j=j+1 end end
114       push(tmp,now) end
115     return #tmp==#b4 and b4 or merge(tmp)
116   end
117   function div(xys,dull,tiny, range,ranges)
118     range = RANGE(i,xys[1].x)
119     ranges = {range}
120     for k,xy in pairs(sort(xys, function(a,b) return a.x < b.x end)) do
121       if k < #xys - tiny and xy.x ~= xys[k+1].x and
122         range.has.n > tiny and range.hi - range.lo > dull
123       then range = push(ranges, RANGE(i, range.hi, xy.x)) end
124       range.hi = xy.x
125       range.has:add(xy.y) end
126     ranges[1].lo = -math.huge
127     ranges[#ranges].hi = math.huge
128     return ranges
129   end
130   xys = {}
131   for _,x in pairs(is) do push(xys, {x=x, y="best"}) end
132   for _,x in pairs(js) do push(xys, {x=x, y="rest"}) end
133   return merge(div(xys, xpects(i,j)*your.dull, (#xys)^your.Tiny)) end
134
135 -----
136
137 function SYM.new(at,s)
138   return new(SYM,{n=0, at=at or 0, txt=s or "", has={}, most=0, mode=nil}) end
139
140 function SYM.add(i,x,count)
141   count = count or 1
142   i.has[x] = count + (i.has[x] or 0)
143   if i.has[x] > i.most then i.most, i.mode = i.has[x], x end
144   return x end
145
146 function SYM.dist(i,x,y) return x=="?" and y=="?" and 1 or x==y and 0 or 1 end
147
148 function SYM.div(i, e)
149   e=0; for _,n in pairs(i.has) do e=e-n/i.n*math.log(n/i.n,2) end; return e end
150
151 function SYM.merged(i,j, k)
152   k= SYM(i.at, i.txt)
153   for x,count in pairs(i.has) do k:add(x,count) end
154   for x,count in pairs(j.has) do k:add(x,count) end
155   return k end
156
157 function SYM.mid(i) return i.mode end
158
159 function SYM.ranges(i,j, ranges,t,n,xpect)
160   t,ranges = {},{}
161   for x,n in pairs(i.has) do t[x]=t[x] or SYM(); t[x]:add("best",n) end
162   for x,n in pairs(j.has) do t[x]=t[x] or SYM(); t[x]:add("rest",n) end
163   for x,stats in pairs(t) do push(ranges, RANGE(i,x)) end
164   return ranges end
165
166 -----
167
168 function EG.new(t) return new(EG, {cooked={}, has=t}) end
169
170 function EG.better(eg1,eg2,egs)
171   local s1,s2,e,n,a,b = 0,0,10,#egs.cols.y
172   for _,col in pairs(egs.cols.y) do
173     a = col:norm(eg1.has[col.at])
174     b = col:norm(eg2.has[col.at])
175     s1 = s1 - e^(col.w * (a-b)/n)
176     s2 = s2 - e^(col.w * (b-a)/n) end
177   return s1/n < s2/n end
178
179 function EG.cols(i,cols) return map(cols,function(x) return i.has[x.at] end) end
180
181 function EG.dist(i,j,egs, a,b,d,n)
182   d,n = 0, #egs.cols.x + 1E-31
183   for _,col in pairs(egs.cols.x) do
184     a,b = i.has[col.at], j.has[col.at]
185     d = d + col:dist(a,b) ^ your.p end
186   return (d/n) ^ (1/your.p) end
187
188

```

```

184 -----
185 function RANGE.new(col,lo,hi,has)
186   lo = lo or -math.huge
187   return new(RANGE, {score=nil,col=col, lo=lo, hi=hi or lo, has=has or SYM()}) e
188 nd
189
190 function RANGE.__tostring(i)
191   if i.lo == i.hi then return fmt("%s==%s",i.col.txt,i.lo) end
192   if i.lo == -math.huge then return fmt("%s<%s",i.col.txt,i.hi) end
193   if i.ho == math.huge then return fmt("%s>=%s",i.col.txt,i.lo) end
194   return fmt("%s<=%s<%s", i.col.txt, i.lo, i.hi) end
195
196 function RANGE.select(i,eg, x)
197   x = eg.has[i.col.at]
198   return x=="?" or i.lo <= x and x < i.hi end
199
200 function RANGE.merge(i,j, k)
201   k = RANGE(i.col, i.lo, j.hi, i.has:merged(j.has))
202   if k.has:div()*1.01 <= xpects(i.has, j.has) then return k end end
203
204 function RANGE.eval(i,goal)
205   local best, rest, goals = 0,0,{}
206   if not i.score then
207     function goals.smile(b,r) return r>b and 0 or b*b/(b+r +1E-31) end
208     function goals.frown(b,r) return b<r and 0 or r*r/(b+r +1E-31) end
209     function goals.xplor(b,r) return 1/(b+r +1E-31) end
210     function goals.doubt(b,r) return 1/(math.abs(b-r) +1E-31) end
211     for x,n in pairs(i.has) do
212       if x==goal then best = best+n/i.n else rest = rest+n/i.n end end
213     i.score = best + rest < 0.01 and 0 or goals[your.goal](best,rest) end
214     return i.score end
215
216 -----
217 function COLS.new(eg, i,now,where)
218   i = new(COLS,{all={}, x={}, y={}})
219   for at,s in pairs(eg) do -- First row. Create the right columns
220     now = push(i.all, (s:find^[A-Z]" and NUM or SYM) (at,s))
221     where = (s:find"- " or s:find"+") and i.y or i.x
222     if not s:find"." then push(where, now) end end
223     return i end
224
225 function COLS.add(i,eg)
226   return map(i.all, function(col) return col:add(eg[col.at]) end) end
227
228 -----
229 function EGS.new(i) return new(EGS, {rows={}, cols=nil}) end
230
231 function EGS.add(i,eg)
232   eg = eg.has and eg.has or eg -- If eg has data buried inside, expose it.
233   if i.cols then push(i.rows,EG(i.cols:add(eg))) else i.cols=COLS(eg) end end
234
235 function EGS.clone(i,init, j)
236   j = EGS()
237   j:add(map(i.cols.all, function(col) return col.txt end))
238   for _,x in pairs(init) do j:add(x) end
239   return j end
240
241 function EGS.cluster(i, rows)
242   local zero,one,two,ones,twos,both,a,b,c
243   zero = any(rows)
244   one = i:far(zero)
245   two,c = i:far(one)
246   ones,twos,both = i:clone(), i:clone(),{}
247   for _,eg in pairs(rows) do
248     a = eg:dist(one, i)
249     b = eg:dist(two, i)
250     push(both, ((a^2 + c^2 - b^2) / (2*c),eg)) end
251   (n <= #both//2 and ones or twos):add(pair[2]) end
252   if your.better and two:better(one,i) then ones,twos=twos,ones end
253   return ones, twos end
254
255 function EGS.splitter(i,top)
256   local one, two = top:cluster(i.rows)
257   local out = {}
258   for n,x1 in pairs(one.cols.x) do
259     local x2, fun, ranges = two.cols.x[n]
260     rows = function(row) local z=row.has[x1.at]; if z=="?" then return z end end
261     ranges = x1:ranges(x2, map(one.rows, rows), map(two.rows, rows))
262     push(out, {xpects(map(ranges,function(r) return x.has end)), ranges}) end
263   return lefts, firsts, sort(out, firsts)[1] end
264
265 function EGS.far(i,egl, fun,tmp)
266   fun = function(eg2) return {eg2, egl:dist(eg2,i)} end
267   tmp = #i.rows > your.Some and any(i.rows, your.Some) or i.rows
268   tmp = sort(map(tmp, fun), seconds)
269   return table.unpack(tmp[#tmp*your.far//1] ) end
270
271 function EGS.from(t, i)
272   i=i or EGS(); for _,eg in pairs(t) do i:add(eg) end; return i end
273
274 function EGS.mid(i,cols)
275   return map(cols or i.all, function(col) return col:mid() end) end
276
277 function EGS.read(file, i)
278   i=i or EGS(); for eg in csv(file) do i:add(eg) end; return i end
279
280
281 -----
282 function any(t, n)
283   if not n then return t[randi(1,#t)] end
284   u={};for j=1,n do push(u,any(t)) end; return u end
285
286 our.fails = 0
287 function asserts(test,msg)
288   msg=msg or ""
289   if test then return print(" PASS:".msg) end
290   our.fails = our.fails+1
291   print(" FAIL:".msg)
292   if your.Debug then assert(test,msg) end end
293
294 function coerce(x)
295   if x=="true" then return true elseif x=="false" then return false end
296   return tonumber(x) or x end
297
298 function copy(t,u)
299   u={}; for k,v in pairs(t) do u[k]=v end
300   return setmetatable(u, getmetatable(t)) end
301
302 function csv(file, x,row)
303   function row(x, t)
304     for y in x:gsub("%s+", ""):gmatch("[^,]+") do push(t,coerce(y)) end
305     return t
306   end
307   file = io.input(file)
308   return function()
309     x=io.read(); if x then return row(x, {}) else io.close(file) end end end
310
311 function defaults(help_string, t,fun)
312   function fun(flag,x)
313     for n,txt in ipairs(arg) do
314       if txt:sub(1,1)=="-" and flag:match("^"..txt:sub(2).."*")
315       then x = x=="false" and"true" or x=="true" and"false" or arg[n+1] end end
316       t[flag] = coerce(x)
317     end
318     t = {}
319     help_string:gsub("\n [-](^%s+)[^%n]*%s([%s]+)", fun)
320     return t end
321
322 function firsts(a,b) return a[1] < b[1] end
323
324 function fmt(...) return string.format(...) end
325
326 function main(our,your, reset)
327   our.defaults = defaults(our.help)
328   reset = function() for k,v in pairs(our.defaults) do your[k] = v end end
329   reset()
330   if your.h then
331     print(our.help)
332   else
333     our.fails = 0
334     for _,one in pairs(your.todo=="all" and slots(our.go) or {your.todo}) do
335       reset()
336       our.go[one]() end
337     for k,v in pairs(_ENV) do
338       if not our.b4[k] then print("?rogues",k,type(v)) end end
339     return our.fails end end
340
341 function map(t,f, u)
342   u = {};for k,v in pairs(t) do push(u,(f or same)(v)) end; return u end
343
344 our.oid=0
345 function new(mt,x)
346   our.oid = our.oid+1; x._oid = our.oid -- Everyone gets a unique id.
347   return setmetatable(x,mt) end -- Methods now delegate to 'mt'.
348
349 function o(t)
350   local u,key
351   key= function(k) return fmt(":%s%s", k, o(t[k])) end
352   if type(t) ~= "table" then return tostring(t) end
353   u = #t>0 and map(t,o) or map(slots(t),key)
354   return (t._is or "").."["..table.concat(u, " ").".."]" end
355
356 function push(t,x) table.insert(t,x); return x end
357
358 your.seed = your.seed or 10019
359 function rand(lo,hi)
360   your.seed = (16807 * your.seed) % 2147483647
361   return (lo or 0) + ((hi or 1) - (lo or 0)) * your.seed / 2147483647 end
362
363 function randi(lo,hi) return math.floor(0.5 + rand(lo,hi)) end
364
365 function rnd(x,d, n)
366   if type(x)=="number" then return x end
367   n=10*(d or your.round)
368   return math.floor(x*n+0.5)/n end
369
370 function rnds(t,d) return map(t,function(x) return rnd(x,d) end) end
371
372 function same(x,...) return x end
373
374 function seconds(a,b) return a[2] < b[2] end
375
376 function slots(t, u)
377   u={}
378   for k,_ in pairs(t) do if tostring(k):sub(1,1) ~= "-" then push(u,k) end end
379   return sort(u) end
380
381 function sort(t,f) table.sort(t,f); return t end
382
383 function xpects(t)
384   local sum,n = 0,0
385   for _,z in pairs(t) do n = n + z.n; sum = sum + z.n*z:div() end
386   return sum/n end
387

```

```

387 -----
388 our.go={ }    -- list of enabled tests
389 our.nogo={ }  -- list of disabled test
390 local go, nogo = our.go,our.nogo
391
392 function go.settings()
393   print("our",o(our))
394   print("your",o(your)) end
395
396 function go.range( r)
397   r=RANGE(NUM(10,"fred"),"apple")
398   assert(tostring(r) == "fred == apple", "print ok") end
399
400 function go.num( m,n)
401   m=NUM(); for j=1,10 do m:add(j) end
402   n=copy(m); for j=1,10 do n:add(j) end
403   asserts(2.95 == rnd(n:div()),"sd ok") end
404
405 function go.egs( egs)
406   egs = EGS.read(your.file)
407   asserts(egs.cols.y[1].hi==5140,"most seen") end
408
409 function go.clone( egs1,egs2,s1,s2)
410   egs1 = EGS.read(your.file)
411   s1 = o(egs1.cols.y)
412   egs2 = egs1:clone(egs1.rows)
413   s2 = o(egs2.cols.y)
414   asserts(s1==s2, "cloning works") end
415
416 function go.dist()
417   local egs,egl,dist,tmp,j1,j2,d1,d2,d3,one
418   egs = EGS.read(your.file)
419   egl = egs.rows[1]
420   dist = function(eg2) return {eg2,egl:dist(eg2,egs)} end
421   tmp = sort(map(egs.rows, dist), seconds)
422   one = tmp[1][1]
423   for j=1,10 do
424     j1 = randi(1,#tmp)
425     j2 = randi(1,#tmp)
426     if j1>j2 then j1,j2=j2,j1 end
427     d1 = tmp[j1][1]:dist(one,egs)
428     d2 = tmp[j2][1]:dist(one,egs)
429     asserts(d1 <= d2,"distance") end end
430
431 function go.cluster( top,left,right)
432   top = EGS.read(your.file)
433   left, right = top:cluster()
434   for n,t in pairs{top,left,right} do print(n,o(rnds(t:mid(t.cols.y)))) end
435 end
436
437 os.exit( main(our,your) )

```