

```

1  -----
2  ---
3  ---
4  ---
5  ---
6  ---
7  ---
8  ---
9  ---
10 ---
11 ---
12 ---
13 ---
14 ---
15 ---
16 ---
17 ---
18 ---
19 ---
20 ---
21 ---
22 ---
23 ---
24 ---
25 local b4={}; for k,_ in pairs(_ENV) do b4[k]=k end
26 local the,help={},{{
27
28 lua l5.lua [OPTIONS]
29 L5 == a very little LUA learning lab
30 (c)2022, Tim Menzies, BSD 2-clause license
31
32 OPTIONS (for changing the inference):
33
34 -cohen -c F cohen's small effect size      = .35
35 -far   -F F look no further than "far"      = .9
36 -keep  -k k items to keep in a number      = 512
37 -leaves -l l leaf size                     = .5
38 -p     -p P distance calcs coefficient      = 2
39 -seed  -S P random number seed             = 10019
40 -some  -s s look only at "some" items       = 512
41
42 OPTIONS (for housekeeping):
43
44 -dump  -d d exit on error, with stacktrace = false
45 -file  -f S where to get data               = ../etc/data/auto93.csv
46 -help  -h h show help                      = false
47 -rnd   -r S format string                   = %5.2f
48 -todo  -t S start-up action                 = nothing
49
50 KEY: S=string, P=poisint, F=float
51 }}
52
53 local as,o = setmetatable
54 local function obj( t )
55   t={__tostring=o}; t.__index=t
56   return as(t, {__call=function( _,... ) return t.new( _,... ) end}) end
57
58 ---
59 ---
60 ---
61 ---
62 ---
63 local Sym = obj() -- Where to summarize symbols
64 function Sym:new(at,s) return as({
65   is="Sym", -- type
66   at=at or 0, -- column index
67   name=s or "", -- column name
68   n=0, -- number of items summarized in this column
69   all={}, -- all[x] = n means we've seen "n" repeats of "x"
70   most=0, -- count of the most frequently seen symbol
71   mode=nil -- the most commonly seen letter
72 }, Sym) end
73
74 local Num = obj() -- Where to summarize numbers
75 function Num:new(at,s) return as({
76   is="Num", -- type
77   at=at or 0, -- column index
78   name=s or "", -- column name
79   n=0, -- number of items summarizes in this column
80   mu=0, -- mean (updated incrementally)
81   m2=0, -- second moment (updated incrementally)
82   sd=0, -- standard deviation
83   all={}, -- a sample of items seen so far
84   lo=1E31, -- lowest number seen
85   hi=-1E31, -- highest number seen
86   w=(s or ""):find"-$" and -1 or 1 -- "-1"= minimize and "1"= maximize
87 }, Num) end
88
89 local Egs = obj() -- Where to store examples, summarized into Syms or Nums
90 function Egs:new(names, i,col,here) i=as({
91   is="Egs", -- type
92   all={}, -- all the rows
93   names=names, -- list of name
94   cols={}, -- list of all columns (Nums or Syms)
95   x={}, -- independent columns (nothing marked as "skip")
96   y={}, -- dependent columns (nothing marked as "skip")
97 }, Egs)
98 for at,name in pairs(names) do
99   col = (name:find"^[A-Z]" and Num or Sym) (at,name)
100   i.cols[i+1].cols = col
101   here = name:find"[-+]" and i.y or i.x
102   if not name:find"$*" then here[i+1 + #here] = col end end
103 return i end
104
105 ---
106 ---
107 ---
108 function Num.clone(i) return Num(i.at, i.name) end
109 function Sym.clone(i) return Sym(i.at, i.name) end
110
111 local data
112 function Egs.clone(i, rows, copy)
113   copy = Egs(i.names)
114   for _,row in pairs(rows or {}) do data(copy,row) end
115   return copy end
116
117 --[[
118 ## Coding Conventions
119 - "I" not "self"
120 - if something holds a list of thing, name the holding variable "all"
121 - no inheritance
122 - only define a method if that is for polymorphism
123 - when you can, write functions down on one line
124 - all config items into a global "the" variable
125 - all the test cases (or demos) are "function Demo.xxx".
126 - random seed reset so carefully, just once, at the end of the code.
127 - usually, no line with just "end" on it
128 ]]
```

```

129 ---
130 ---
131 ---
132 ---
133 ---
134 ---
135 local r = math.random
136 local fmt = string.format
137 local unpack = table.unpack
138 local function push(t,x) table.insert(t,x); return x end
139
140 ---
141 ---
142 ---
143 local thing,things,file2things
144 function thing(x)
145   x = x:match"^(%s*)(-)%s*$"
146   if x=="true" then return true elseif x=="false" then return false end
147   return tonumber(x) or x end
148
149 function things(x,sep, t)
150   t={} for y in x:gmatch(sep or"([^\+]+)" do push(t,thing(y)) end
151   return t end
152
153 function file2things(file, x)
154   file = io.input(file)
155   return function()
156     x=io.read();
157     if x then return things(x) else io.close(file) end end end
158
159 ---
160 ---
161 ---
162 local last,per,any,many
163 function last(a) return a[ #a ] end
164 function per(a,p) return a[ (p*#a)//1 ] end
165 function any(a) return a[ math.random(#a) ] end
166 function many(a,n, u) u={}; for j=1,n do push(u,any(a)) end; return u end
167
168 ---
169 ---
170 ---
171 local firsts,sort,map,slots
172 function firsts(a,b) return a[1] < b[1] end
173 function sort(t,f) table.sort(t,f); return t end
174 function map(t,f, u) u={};for k,v in pairs(t) do push(u,f(v)) end; return u end
175 function slots(t, u,s)
176   u={}
177   for k,v in pairs(t) do s=tostring(k);if s:sub(1,1)~="_" then push(u,k) end end
178   return sort(u) end
179
180 ---
181 ---
182 ---
183 local oo,rnd, rnds -- local o was declared above (in "new")
184 function oo(t) print(o(t)) end
185 function o(t,seen, key,xseen,u)
186   seen = seen or {}
187   if type(t)~="table" then return tostring(t) end
188   if seen[t] then return "..." end
189   seen[t] = t
190   key = function(k) return fmt(":%s %s",k,o(t[k],seen)) end
191   xseen = function(x) return o(x,seen) end
192   u = #t>0 and map(t,xseen) or map(slots(t),key)
193   return (t.is or "")..'{'..table.concat(u," ")...'}' end
194
195 function rnds(t,f) return map(t, function(x) return rnd(x,f) end) end
196 function rnd(x,f)
197   return fmt(type(x)=="number" and (x~x//1 and f or the.rnd) or "%s",x) end
198
199 ---
200 ---
201 ---
202 local Demo, ok = {fails=0}
203 function ok(test,msg)
204   print(test and "PASS:" or "FAIL:",msg or "")
205   if not test then
206     Demo.fails=Demo.fails+1
207     if the.dump then assert(test,msg) end end end
208
209 function Demo.main(todo,seed)
210   for k,one in pairs(todo=="all" and slots(Demo) or {todo}) do
211     if k ~="main" and type(Demo[one]) == "function" then
212       math.randomseed(seed)
213       Demo[one]() end end
214   for k,v in pairs(_ENV) do if not b4[k] then print("?",k,type(v)) end end
215   return Demo.fails end
216
217 local function settings(txt, d)
218   d={}
219   txt:gsub("un ([-]|([%s+))[%s]+(-|[%s+])^un)*%s([%s+])",
220   function(long,key,short,x)
221     for n,flag in ipairs(arg) do
222       if flag==short or flag==long then
223         x = x=="false" and true or x=="true" and "false" or arg[n+1] end end
224     if x=="false" then the[key]=false elseif x=="true" then the[key]=true else
225       d[key] = tonumber(x) or x end end)
226   if d.help then print(help) end
227   return d end
```

UPDATE COLS

```
228 ---
229 ---
230 ---
231 ---
232 local add
233 function add(i,x, inc)
234   inc = inc or 1
235   if x ~= "?" then
236     i.n = i.n + inc
237     i:internalAdd(x,inc) end
238   return x end
239
240 function Sym.internalAdd(i,x,inc)
241   i.all[x] = inc + (i.all[x] or 0)
242   if i.all[x] > i.most then i.most, i.mode = i.all[x], x end end
243
244 function Num.internalAdd(i,x,inc, d)
245   for j=1,inc do
246     d = x - i.mu
247     i.mu = i.mu + d/i.n
248     i.m2 = i.m2 + d*(x - i.mu)
249     i.sd = (i.m2<0 or i.n<2) and 0 or ((i.m2/(i.n-1))^0.5)
250     i.lo = math.min(x, i.lo)
251     i.hi = math.max(x, i.hi)
252     if #i.all < the.keep then push(i.all,x)
253     elseif r() < they.keep/i.n then i.all[r(#i.all)]=x end end end
254
255 ---
256 ---
257 ---
```

MAKE DATA

```
258 local file2Egs -- not "local data" (since defined above)
259 function data(i,row)
260   push(i.all, row)
261   for _,col in pairs(i.cols) do add(col, row[col.at]) end
262   return i end
263
264 function file2Egs(file, i)
265   for row in file2things(file) do
266     if i then data(i,row) else i = Egs(row) end end
267   return i end
268
269 ---
270 ---
271 ---
272 function Sym.mid(i) return i.mode end
273 function Num.mid(i) return i.mu end
274
275 function Num.div(i) return i.sd end
276 function Sym.div(i, e)
277   e=0; for _,n in pairs(i.all) do e=e + n/i.n*math.log(n/i.n,2) end
278   return -e end
279
280 function Egs.mid(i,cols)
281   return map(cols or i.y,function(col) return col:mid() end) end
282
283 local mids
284 function mids(i,rows,cols, seen,tmp,j)
285   j = i:clone()
286   for _,row in pairs(rows) do data(j, row) end
287   return rnds(j:mid(cols)) end
288
289 ---
290 ---
291 ---
```

DISTANCE

```
292 local far,furthest,neighbors,dist
293 function far(i,r1,rows,far)
294   return per(neighbors(i,r1,rows),far or the.far)[2] end
295
296 function furthest(i,r1,rows)
297   return last(neighbors(i,r1,rows))[2] end
298
299 function neighbors(i,r1,rows)
300   return sort(map(rows, function(r2) return {dist(i,r1,r2),r2} end),firsts) end
301
302 function dist(i,row1,row2, d,n,a,b,inc)
303   d,n = 0,0
304   for _,col in pairs(i.x) do
305     a,b = row1[col.at], row2[col.at]
306     inc = a=="?" and b=="?" and 1 or col:dist1(a,b)
307     d = d + inc^the.p
308     n = n + 1 end
309   return (d/n)^(1/the.p) end
310
311 function Sym.dist1(i,a,b) return a==b and 0 or 1 end
312
313 function Num.dist1(i,a,b)
314   if a=="?" then b=i:norm(b); a=b<.5 and 1 or 0
315   elseif b=="?" then a=i:norm(a); b=a<.5 and 1 or 0
316   else a,b = i:norm(a), i:norm(b) end
317   return math.abs(a - b) end
318
319 function Num.norm(i,x)
320   return i.hi - i.lo < 1E-32 and 0 or (x - i.lo)/(i.hi - i.lo) end
321
322 ---
323 ---
324 ---
```

CLUSTER

```
325 local half, cluster, clusters
326 function half(i, rows, project,row,some,east,west,easts,wests,c,mid)
327   function project(row,a,b)
328     a = dist(i,east,row)
329     b = dist(i,west,row)
330     return ((a^2 + c^2 - b^2)/(2*c), row)
331   end
332   some = many(rows, the.some)
333   east = furthest(i,any(some), some)
334   west = furthest(i,east, some)
335   c = dist(i,east,west)
336   easts,wests = {},{}
337   for n, xrow in pairs(sort(map(rows,project),firsts)) do
338     row = xrow[1]
339     if n==#rows//2 then mid=row end
340     push(n <= #rows//2 and easts or wests, row) end
341   return easts, wests, east, west, mid end
342
343 function cluster(i,rows, here,lefts,rights)
344   rows = rows or i.all
345   here = {all=rows}
346   if #rows > 2*(#i.all)^the.leaves then
347     lefts, rights = half(i, rows)
348     if #lefts < #rows then
349       here.lefts = cluster(i,lefts)
350       here.rights = cluster(i,rights) end end
351   return here end
352
353 function clusters(i,t,pre)
354   pre = pre or ""
355   if t then
356     if not t.lefts and not t.rights then
357       print(fmt("%5s%-20s",#t.all, pre), o(mids(i,t.all)))
358     else
359       print(fmt("%5s%-20s",#t.all, pre))
360       clusters(i,t.lefts, "... pre)
361       clusters(i,t.rights, "... pre) end end end
```

DISCRETIZE

```
362 ---
363 ---
364 ---
365 ---
366 local merge,merged,spans,bestSpan
367 function Sym.spans(i, j)
368   local xys,all,one,last,x,y,n = {},{}
369   for x,n in pairs(i.all) do push(xys, {x,"easts",n}) end
370   for x,n in pairs(j.all) do push(xys, {x,"wests",n}) end
371   for _,tmp in ipairs(sort(xys,firsts)) do
372     x,y,n = unpack(tmp)
373     if x ~= last then
374       last = x
375       one = push(all, {lo=x, hi=x, all=Sym(i.at,i.name)}) end
376     add(one.all, y, n) end
377   return all end
378
379 function Num.spans(i, j)
380   local xys,all,lo,hi,gap,one,x,y,n = {},{}
381   lo,hi = math.min(i.lo, j.lo), math.max(i.hi, j.hi)
382   gap = (hi - lo) / (6/the.cohen)
383   for _,n in pairs(i.all) do push(xys, {n,"easts",1}) end
384   for _,n in pairs(j.all) do push(xys, {n,"wests",1}) end
385   one = {lo=lo, hi=lo, all=Sym(i.at,i.name)}
386   all = {one}
387   for _,tmp in ipairs(sort(xys,firsts)) do
388     x,y,n = unpack(tmp)
389     if one.hi - one.lo > gap then
390       one = push(all, {lo=one.hi, hi=x, all=one.all:clone()})
391     end
392     one.hi = x
393     add(one.all, y, n) end
394   all
395   all[1].lo = -math.huge
396   all[#all].hi = math.huge
397   return all end
398
399 function merge(b4, j,n,now,a,b,both)
400   j, n, now = 0, #b4, {}
401   while j < #b4 do
402     j = j+1
403     a, b = b4[j], b4[j+1]
404     if b then
405       both = a.all:merge(b.all)
406       if both then
407         a = {lo=a.lo, hi=b.hi, all=both}
408         j = j + 1 end end
409     push(now,a) end
410   return #now == #b4 and b4 or merge(now) end
411
412 function Sym.merge(i,j, k,ei,ej,ek)
413   k = i:clone()
414   for x,n in pairs(i.all) do add(k,x,n) end
415   for x,n in pairs(j.all) do add(k,x,n) end
416   ei, ej, ek = i:div(), j:div(), k:div()
417   if ek*.99 <= (i.n*ei + j.n*ej)/k.n then
418     return k end end
419
420 function spans(egs1,egs2, spans,tmp,coll,col2)
421   spans = {}
422   for c,coll in pairs(egs1.x) do
423     col2 = egs2.x[c]
424     tmp = coll:spans(col2)
425     if #tmp> 1 then
426       for _,one in pairs(tmp) do push(spans,one) end end end
427   return spans end
428
429 function bestSpan(spans)
430   local divs,ns,n,div,stats,dist2heaven = Num(), Num()
431   function dist2heaven(s) return {(1 - n(s))^2 + (0 - div(s))^2^.5,s} end
432   function div(s) return divs:norm( s.all:div() ) end
433   function n(s) return ns:norm( s.all.n ) end
434   for _,s in pairs(spans) do
435     add(divs, s.all:div())
436     add(ns, s.all.n) end
437   return sort(map(spans, dist2heaven), firsts)[1][2] end
```

```

438
439 function selects(span,row,    lo,hi,at,x)
440   lo, hi, at = span.lo, span.hi, span.all.at
441   x = row[at]
442   if x=="0" then return true end
443   if lo==hi then return x==lo else return lo <= x and x < hi end end
444
445 ---
446 ---  EXPLAIN
447 ---
448
449 function xplain(i,rows,    here,lefs,rights)
450   rows = rows or i.all
451   here = {all=rows}
452   stop = (#i.all)^the.leaves
453   if #rows > stop then
454     lefs0, rights0 = half(i, rows)
455     here.selector = bestSpan(spans(i:clone(lefs0),i:clone(rights0)))
456     lefs1,rights1 = {},{}
457     if #lefs < #rows then
458       for _,row in pairs(rows) do
459         push(selects(here.selector, row) and lefs1 or rights1,row) end
460       here.lefs = xplain(i,lefs1)
461       here.rights = xplain(i,cwricwricwri,cwr:WQ
462     ilefs1)
463     here.lefs = xplain(i,lefs)
464     here.rights= xplain(i,rights) end end
465   return here end
466
467 function clusters(i,t,pre)
468   pre = pre or ""
469   if t then
470     if not t.lefs and not t.rights then
471       print(fmt("%5s%-20s",#t.all, pre), o(mids(i,t.all)))
472     else
473       print(fmt("%5s%-20s",#t.all, pre))
474       clusters(i,t.lefs,  "%. ".. pre)
475     ---
476     ---
477     ---  INVENTION
478     ---
479     ---
480
481 function Demo.the() oo(the) end
482
483 function Demo.many(a)
484   a={1,2,3,4,5,6,7,8,9,10}; ok("{1023}" == o(many(a,3)), "manys") end
485
486 function Demo.egs()
487   ok(5140==file2Egs(the.file).y[1].hi,"reading") end
488
489 function Demo.dist(i)
490   i = file2Egs(the.file)
491   for n,row in pairs(i.all) do print(n,dist(i, i.all[1], row)) end end
492
493 function Demo.far( i,j,row1,row2,row3,d3,d9)
494   i = file2Egs(the.file)
495   for j=1,10 do
496     row1 = any(i.all)
497     row2 = far(i,row1, i.all, .9)
498     d9 = dist(i,row1,row2)
499     row3 = far(i,row1, i.all, .3)
500     d3 = dist(i,row1,row3)
501     ok(d3 < d9, "closer far") end end
502
503 function Demo.half( i,easts,wests)
504   i = file2Egs(the.file)
505   easts,wests = half(i, i.all)
506   oo(mids(i.y, easts))
507   oo(mids(i.y, wests)) end
508
509 function Demo.cluster( i)
510   i = file2Egs(the.file)
511   clusters(i,cluster(i)) end
512
513 function Demo.spans( i,j,tmp,easts,wests)
514   i = file2Egs(the.file)
515   easts, wests = half(i, i.all)
516   oo(bestSpan(spans(i:clone(easts), i:clone(wests)))) end
517
518 -----
519 the = settings(help)
520 Demo.main(the.todo, the.seed)

```