

```

1  #!/usr/bin/env lua
2  -- vi: filetype=lua :
3  -----
4  -----
5  -----
6  -----
7  -----
8  -----
9  -----
10 -----
11
12 -- BSD 2-Clause License
13 -- Copyright (c) 2022, Tim Menzies
14
15 -- Redistribution and use in source and binary forms, with or without
16 -- modification, are permitted provided that the following conditions are met:
17
18 -- 1. Redistributions of source code must retain the above copyright notice, this
19 --    list of conditions and the following disclaimer.
20
21 -- 2. Redistributions in binary form must reproduce the above copyright notice,
22 --    this list of conditions and the following disclaimer in the documentation
23 --    and/or other materials provided with the distribution.
24
25 -- THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
26 -- AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
27 -- IMPLIED WARRANTIES OF MERCHANTABILITY & FITNESS FOR A PARTICULAR PURPOSE ARE
28 -- DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
29 -- FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
30 -- DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
31 -- SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
32 -- CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
33 -- OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
34 -- OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
35
36 local b4={}; for k, _ in pairs(_ENV) do b4[k]=k end
37 local help={}
38 brknbad.lua: explore the world better, explore the world for good.
39 (c) 2022, Tim Menzies
40
41
42      Ba      Bad <----- planning= (better - bad)
43      56      |      monitor = (bad - better)
44      -----|----- v
45      Be      4      Better
46      -----|-----
47
48 USAGE:
49 ./bnb [OPTIONS]
50
51 OPTIONS:
52 -bins -b      max. number of bins      = 16
53 -cohen -c      cohen                    = .35
54 -goal -g      goal                      = recurrence-events
55 -K -K          manage low class counts   = 1
56 -M -M          manage low evidence counts = 2
57 -seed -S      seed                      = 10019
58 -wait -w      wait                      = 10
59
60 OPTIONS (other):
61 -dump -d      dump stack on error, then exit = false
62 -file -f      file name                  = ../etc/data/breastcancer.csv
63 -help -h      show help                  = false
64 -todo -t      start up action             = nothing
65
66 ]]
67
68 local ent,per
69 local push,map,collect,copy,powerset
70 local sort,upl,upx,downl,slots,upl,downl
71 local words,thing,things,items
72 local cli
73 local fmt,o,oo
74 local inc,inc2,inc3,has,has2,has3
75 local ok,ish,roques
76 local cols,update,classify,test,train,score,nbl,nb2,abcd
77 local bins,nb3
78 local eg,the,ako={},{},{}
79
80 --- column types
81 ---
82
83 local ako={}
84 ako.num = function(x) return x:find("[A-Z]") end
85 ako.goal = function(x) return x:find("[+!]" end
86 ako.klass = function(x) return x:find("$" end
87 ako.ignore = function(x) return x:find("$" end
88 ako.weight = function(x) return x:find"$" and -1 and 1 end
89
90 --- structs
91 ---
92
93 local it={}
94 function it.num()
95     return {nump=true, n=0, at=0, txt="",lo=1E32, hi=-1E32, mu=0, bins={}} end
96
97 function it.sym()
98     return {nump=false, n=0, at=0, txt="", has={}, most=0, mode=nil} end
99
100 function it.three()
101     return {all={}, nums={}, syms={}} end
102
103 function it.cols()
104     return {names={}, klass=nil,xy= it.three(), x= it.three(), y= it.three()} end
105
106 function it.egs()
107     return {h={}, nh=0, e={}, ames=nil, n=0, bests=0, rests=0,
108             best={}, rest={}, log={}, cols=nil} end
109
110 -----
111 --- BASIC
112 ---
113
114 function classify(i,t)
115     local hi,out = -1
116     for h, _ in pairs(i.h) do
117         local prior = ((i.h[h] or 0) + the.K)/(i.n + the.K*i.nh)
118         local l = prior
119         for col,x in pairs(t) do
120             if x ~= "" and col ~= #t then
121                 l=l+(has3(i.e,col,x,h) + the.M*prior)/((i.h[h] or 0) + the.M) end end
122             if l>hi then hi,out=l,h end end
123         return out end
124
125 function test(i,t)
126     if i.n > the.wait then push(i.log,{want=t[#t], got=classify(i,t)}) end end
127
128 function train(i,t)
129     local more, kl = false, t[#t]
130     for col,x in pairs(t) do
131         if x ~= "?" then
132             more = true
133             inc3(i.e, col, x, kl)
134             if col ~= #t then
135                 inc2(kl==the.goal and i.best or i.rest, col,x) end end end
136         if more then
137             i.n = i.n + 1
138             if not i.h[kl] then i.nh = i.nh + 1 end
139             inc(i.h, kl)
140             if kl==the.goal then i.bests=i.bests+1 else i.rests=i.rests+1 end end end
141         return out end
142
143 function score(i)
144     local acc,out=0,{}
145     for _,x in pairs(i.log) do if x.want==x.got then acc=acc+1/#i.log end end
146     for col,xns in pairs(i.best) do
147         for x,b in pairs(xns) do
148             local r1 = has2(i.rest,col,x)/i.rests
149             local b1 = b/i.bests
150             push(out, {100*(b1^2/(b1+r1))/1, col,x,b}) end end
151     return acc, sort(out,downl) end
152
153 function nbl(file, log)
154     local i = {h={}, nh=0,e={}, names=nil, n=0, wait=the.wait,
155               bests=0,rests=0,best={}, rest={},log=log or {}}
156     for row in items(file) do
157         if not i.names then i.names=row else
158             test(i,row); train(i,row) end end
159     return i end
160
161 --- nums and syms
162 ---
163
164 function cols(names)
165     local i = it.cols()
166     local function keep(now, at) -- keep in "all" plus in one of "nums" or "syms"
167         push(ako.num(now.txt) and at.nums or at.syms, push(at.all, now)) end
168     i.names = names
169     for j,txt in pairs(names) do
170         local now = ako.num(txt) and it.num() or it.sym()
171         now.at, now.txt, now.w = j, txt, ako.weight(txt)
172         keep(now, i.xy)
173         if not ako.ignore(txt) then
174             keep(now, ako.goal(txt) and i.y or i.x)
175             if ako.klass(txt) then i.klass=now end end end
176         return i end
177
178 function update(i,t)
179     local function num(col, x)
180         col.mu = col.mu + (x - col.mu)/col.n
181         col.lo = math.min(x, col.lo)
182         col.hi = math.max(x, col.hi) end
183
184     local function num(col, x)
185         col.has[x] = 1 + (col.has[x] or 0)
186         if col.has[x] > col.most then col.mode,col.most = x,col.has[x] end end
187
188     for _,col in pairs(i.cols.xy.all) do
189         local x = t[col.at]
190         if x ~= "?" then
191             col.n = col.n + 1
192             (col.nump and num or sym) (col,x) end end
193         return t end
194
195 --- v v i t h i a v v d
196 ---
197
198 function nb2(file, log)
199     local tmp, i = {}, it.egs()
200     local function discretize(j,x)
201         if x=="?" then
202             col = i.cols.xy.all[j]
203             if col.nump then
204                 x = (x - col.lo) // ((col.hi - col.lo+1E-32) / the.bins) end end
205             return x end
206     -- start
207     for row in items(file) do
208         if not i.cols then i.cols=cols(row) else push(tmp,update(i,row)) end end
209     for _,row in pairs(tmp) do
210         row=collect(row,discretize)
211         test(i,row); train(i,row) end
212     return i end
213

```

```

214 -----
215 == METRICS
216
217
218
219 function abcd(gotwants, show)
220 local i, exists, add, report, pretty
221 i={data=data or "data", rx= rx or "rx", known={}, a={}, b={}, c={}, d={}, yes=0, no=0}
222
223 function exists(x, new)
224 new = not i.known[x]
225 inc(i.known, x)
226 if new then
227 i.a[x]=i.yes + i.no; i.b[x]=0; i.c[x]=0; i.d[x]=0 end end
228
229 function report ( p, out, a, b, c, d, pd, pf, pn, f, acc, g, prec)
230 p = function (z) return math.floor(100*z + 0.5) end
231 out= {}
232 for x, _ in pairs( i.known ) do
233 pd, pf, pn, prec, g, f, acc = 0, 0, 0, 0, 0, 0
234 a= (i.a[x] or 0); b= (i.b[x] or 0); c= (i.c[x] or 0); d= (i.d[x] or 0);
235 if b+d > 0 then pd = d / (b+d) end
236 if a+c > 0 then pf = c / (a+c) end
237 if a+c > 0 then pn = (b+d) / (a+c) end
238 if c+d > 0 then prec = d / (c+d) end
239 if 1-pf+pd > 0 then g=2*(1-pf) * pd / (1-pf+pd) end
240 if prec+pd > 0 then f=2*prec*pd / (prec + pd) end
241 if i.yes + i.no > 0 then
242 acc= i.yes / (i.yes + i.no) end
243 out[x] = {data=i.data, rx=i.rx, num=i.yes+i.no, a=a, b=b, c=c, d=d, acc=p(acc),
244 prec=p(prec), pd=p(pd), pf=p(pf), f=p(f), g=p(g), class=x} end
245 return out end
246
247 function pretty(t)
248 print""
249 local s1 = "%10s| %10s| %4s| %4s| %4s| %4s"
250 local s2 = "| %3s| %3s| %3s| %4s| %3s| %3s|"
251 local d, s = "", (s1 .. s2)
252 print(fmt(s, "db", "rx", "a", "b", "c", "d", "acc", "pd", "pf", "prec", "f", "g"))
253 print(fmt(s, d, d, d, d, d, d, d, d, d, d, d, d))
254 for _, x in pairs(slots(t)) do
255 local u = t[x]
256 print(fmt(s, "%s", u.data, u.rx, u.a, u.b, u.c, u.d,
257 u.acc, u.pd, u.pf, u.prec, u.f, u.g, x)) end end
258
259 -- start
260 for _, one in pairs(gotwants) do
261 exists(one.want)
262 if one.want == one.got then i.yes=i.yes+1 else i.no=i.no+1 end
263 for x, _ in pairs(i.known) do
264 if one.want == x
265 then inc(one.want == one.got and i.d or i.b, x)
266 else inc(one.got == x and i.c or i.a, x) end end end
267 return show and pretty(report()) or report() end

```

```

268 -----
269 == SUPER RANGES
270
271
272
273 function nb3(file, log)
274 local tmp, i = {}, it.egs()
275 local function discretize(j, x, bins)
276 if x ~= "?" then
277 bins = i.nums[j]
278 if bins then
279 for _, bin in pairs(bins) do
280 if bin.lo <= x and x < bin.hi then return bin.id end end end end
281 return x end
282
283 function update1(i, row)
284 update(i, row)
285 -- start
286 for row in items(file) do
287 if not i.cols then i.cols = cols(row) else push(tmp, update1(i, row)) end end
288 for _, col in pairs(i.cols.x.nums) do i.nums[j] = bins(xys, j) end
289 for _, row in pairs(tmp) do
290 row = collect(row, discretize);
291 test(i, row); train(i, row) end
292 return i end
293
294 --- Bind bins
295 ---
296 function bins(xys, ref)
297 xys = sort(xys, upx)
298 local cohen = the.cohen * (per(xys, .9).x - per(xys, .1).x) / 2.56
299 local minItems = #xys / the.bins
300 local out, b4 = {}, -math.huge
301 local function add(f, z) f[z] = (f[z] or 0) + 1 end
302 local function sub(f, z) f[z] = f[z] - 1 end
303 local function argmin(lo, hi)
304 local lhs, rhs, cut, div, xpect, xy = {}, {}
305 for j=lo, hi do add(rhs, xys[j].y) end
306 div = ent(rhs)
307 if hi-lo+1 > 2*minItems
308 then
309 for j=lo, hi - minItems do
310 add(lhs, xys[j].y)
311 sub(rhs, xys[j].y)
312 local n1, n2 = j - lo + 1, hi - j
313 if n1 > minItems and -- enough items (on left)
314 n2 > minItems and -- enough items (on right)
315 xys[j].x == xys[j+1].x and -- there is a break here
316 xys[j].x - xys[lo].x > cohen and -- not trivially small (on left)
317 xys[hi].x - xys[j].x > cohen -- not trivially small (on right)
318 then xpect = (n1*ent(lhs) + n2*ent(rhs)) / (n1+n2)
319 if xpect < div then -- cutting here simplifies things
320 cut, div = j, xpect end end end --end for
321 end -- end if
322 if cut
323 then argmin(lo, cut)
324 argmin(cut+1, hi)
325 else b4 = push(out, {ref=ref, lo=b4, hi=xys[hi].x, n=hi-lo+1, div=div}).hi end
326 end -----
327 argmin(1, #xys)
328 for j, bin in pairs(out) do bin.id = j end
329 out[#out].hi = math.huge
330 return out end
331

```

```

332 -----
333
334 --- XPLAIN
335 ---
336 -----
337
338 ---
339 --- MISC
340 ---
341 --- maths
342 ---
343 ---
344
345 function per(t,p) return t[ (p or .5)*#t//1 ] end
346
347 function ent(t)
348   local n=0; for _,m in pairs(t) do n = n+m end
349   local e=0; for _,m in pairs(t) do if m>0 then e = e+m/n*math.log(m/n,2) end end
350   return -e end
351
352 --- check
353 ---
354
355 function ish(x,y,z) return math.abs(x-y) <= (z or 0.001) end
356
357 local fails=0
358 function ok(test,msg)
359   print("", test and "PASS" or "FAIL ",msg or "")
360   if not test then
361     fails = fails+1
362     if the and the.dump then assert(test,msg) end end end
363
364 function rogues()
365   for k,v in pairs(_ENV) do if not b4[k] then print("??",k,type(v)) end end end
366
367 --- count
368 ---
369
370 function inc(f,a,n) f=f or {};f[a]=(f[a] or 0) + (n or 1) return f end
371 function inc2(f,a,b,n) f=f or {};f[a]=inc(f[a] or {},b,n); return f end
372 function inc3(f,a,b,c,n) f=f or {};f[a]=inc2(f[a] or {},b,c,n);return f end
373
374 function has(f,a) return f[a] or 0 end
375 function has2(f,a,b) return f[a] and has(f[a],b) or 0 end
376 function has3(f,a,b,c) return f[a] and has2(f[a],b,c) or 0 end
377
378 --- lists
379 ---
380
381 function push(t,x) t[1 + #t] = x; return x end
382
383 function map(t, f, u) u={};for k,v in pairs(t) do u[1+#u]=f(v) end;return u end
384 function collect(t,f, u) u={};for k,v in pairs(t) do u[k]=f(k,v)end;return u end
385 function copy(t, u)
386   if type(t) ~= "table" then return t end
387   u={}; for k,v in pairs(t) do u[k]=copy(k) = copy(v) end; return u end
388
389 function powerset(s)
390   local function aux(s)
391     local t = {}
392     for i = 1, #s do
393       for j = 1, #t do
394         t[#t+1] = {s[i],table.unpack(t[j])} end end
395     return t end
396   return sort(aux(s), function(a,b) return #a < #b end) end
397
398 function sort(t,f) table.sort(t,f); return t end
399
400 function upx(a,b) return a.x < b.x end
401 function upl(a,b) return a[1] < b[1] end
402 function downl(a,b) return a[1] > b[1] end
403
404
405 function slots(t, u)
406   local function public(k) return tostring(k):sub(1,1) ~= "." end
407   u={};for k,v in pairs(t) do if public(k) then u[1+#u]=k end end
408   return sort(u) end
409
410 --- string '2 things
411 ---
412 ---
413
414 function words(s,sep, t)
415   sep="([^\n" .. (sep or ".") .. "\n"]+)"
416   t={}; for y in s:gmatch(sep) do t[1+#t] = y end; return t end
417
418 function things(s) return map(words(s), thing) end
419
420 function thing(x)
421   x = x:match("%s*(-)%s*$")
422   if x=="true" then return true elseif x=="false" then return false end
423   return tonumber(x) or x end
424
425 function items(src,f)
426   local function file()
427     src,f = io.input(src),f or things
428     return function() x=io.read();if x then return f(x) else io.close(src) end end
429   end
430   local function tbl( x)
431     x,f = 0, f or function(z) return z end
432     return function() if x< #src then x=x+1; return f(src[x]) end end end
433   return type(src) == "string" and file() or tbl() end
434
435 --- things '2 string
436 ---
437
438 fmt = string.format
439
440 function oo(t) print(o(t)) end
441
442 function o(t, seen, u)
443   if type(t)~="table" then return tostring(t) end
444   seen = seen or {}
445   if seen[t] then return "..." end
446   seen[t] = t
447   local function show1(x) return o(x, seen) end
448   local function show2(k) return fmt("%s %s",k, o(t[k],seen)) end
449   u = #t>0 and map(t,show1) or map(slots(t),show2)
450   return (t.s or "")..."["..table.concat(u, " ").."]" end
451
452 --- cli
453 ---
454
455 function cli(help)
456   local d,used = {},{}
457   help:gsub("\n ([-|!%s+])[%s]+(-[%s+]|^\\n)%s+([%s+])",
458     function(long,k,short,x)
459       assert(not used[short], "repeated short flag ["..short.."]")
460       used[short]=short
461       for n,flag in ipairs(arg) do
462         if flag==short or flag==long then
463           x = x=="false" and true or x=="true" and "false" or arg[n+1] end end
464         d[key] = x==true and true or thing(x) end
465   if d.help then os.exit(print(help)) end
466   return d end

```

467

```

467 -----
468 --- DEMOS
469 ---
470 ---
471 ---
472 function eg.copy(      t,u)
473   t={a={b={c=10},d={e=200}}, f=300}
474   u= copy(t)
475   t.a.b.c= 20
476   print(u.a.b.c)
477   oo(t)
478   oo(u)
479   end
480 ---
481 function eg.collect()
482   local function aux(x,y) return x*y end
483   oo(collect({10,20,30},aux)) end
484 ---
485 function eg.ent()
486   ok(ish(ent{a=9,b=7}, .98886), "entropy") end
487 ---
488 function eg.items()
489   for x in items{10,20,30} do print(x) end
490   local n=0
491   for x in items(the.file) do n=n+1; if n<=5 then oo(x) end end end
492 ---
493 function eg.powerset()
494   for _,x in pairs(powerset{10,20,30,40,50}) do oo(x) end end
495 ---
496 function eg.nb1()
497   local i = nb1(the.file);
498   local acc, out = score(i); print(acc); map(out,oo) end
499 ---
500 -- there is a "?" in the output. nope
501 function eg.nb2()
502   local i = nb2(the.file);
503   local acc, out = score(i); print(acc); map(out,oo) end
504 ---
505 function eg.nb2a()
506   local i = nb2("./etc/data/diabetes.csv");
507   local acc, out = score(i)
508   abcd(i.log, true)
509   map(out,oo) end
510 ---
511 function eg.bins(      t)
512   local t,n = {},30
513   for j=1,n do push(t, {x=j, y=j<.6*n and 1 or j<.8*n and 2 or 3}) end
514   map(bins(t,20),oo)
515   end
516 ---
517 function eg.nb3(      i)
518   print(20)
519   i=nb3("./etc/data/diabetes.csv")
520   for n,bins in pairs(i.nums) do
521     print(n,#bins) end
522   local acc, out = score(i) -- XXX
523   print(#out)
524   print(acc)
525   map(out,oo)
526   end
527

```

```

527 -----
528 --- START
529 ---
530 ---
531 ---
532 fails = 0
533 local defaults=cli(help)
534 local todos = defaults.todo == "all" and slots(eg) or {defaults.todo}
535 for _,todo in pairs(todos) do
536   the = copy(defaults)
537   math.randomseed(the.seed or 10019)
538   if eg[todo] then eg[todo]() end end
539 ---
540 roques()
541 os.exit(fails)
542 ---
543 ---
544 ---
545 ---
546 ---
547 ---
548 ---
549 ---
550 ---
551 ---

```

```

-----
-- [ ] = [ ]
-- [ ] = [ ]
---
###
# = #
#####

```

"This ain't chemistry.  
This is art."