

```

1 local b4={}; for k,v in pairs(_ENV) do b4[k]=v end
2 local any,coerce, csv, fails,fmt,go,id,lt,many,map,obj,push
3 local no,o,oo,ok,per,r,rnd,rnds,sort,sum,the,workl,work
4 local the,help={},{}
5 small: explore the world better, explore the world for good.
6 (c) 2022, Tim Menzies
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172

```

Ba 56 Bad <----- planning= (better - bad)
 monitor = (bad - better)
 Be 4 Better
 v

```

17 USAGE:
18 ./bnb [OPTIONS]
19
20 OPTIONS:
21 -K -K manage low class counts = 1
22 -M -M manage low evidence counts = 2
23 -best -B best set = .5
24 -bins -b max. number of bins = 16
25 -cohen -c cohen = .35
26 -dump -d dump stack+exit on error = false
27 -far -f how far to go for far = .9
28 -file -f file name = ../etc/data/auto93.csv
29 -goal -g goal = recurrence-events
30 -help -h show help = false
31 -leaves -l number of items in leaves = .5
32 -p -p coefficient on distance = 2
33 -rest -R rest is -R*best = 4
34 -rnd -r rounding numbers = %.3f
35 -seed -S seed = 10019
36 -some -s sample size for distances = 512
37 -todo -t start up action = nothing
38 -wait -w wait = 10]]
39
40
41 r = math.random
42 fmt = string.format
43 function lt(x) return function(t,u) return t[x] < u[x] end end
44 function sort(t,f) table.sort(t,type(f)=="string" and lt(f) or f);return t end
45
46 function push(t,x) t[1+#t]=x; return x end
47 function map(t,f, u) u={};for _,v in pairs(t) do u[1+#u]=f(v) end;return u end
48 function sum(t,f, u) u=0; for _,v in pairs(t) do u=u+f(v) end;return u end
49
50 function any(a) return a[r()*(#a)/1] end
51 function many(a,n, u) u={};for j=1,n do push(u,any(a)) end;return u end
52
53 function per(t,p) return t[ ((p or .5)*#t) // 1 ] end
54
55 function oo(t) print(o(t)) end
56
57 function rnds(t,f) return map(t, function(x) return rnd(x,f) end) end
58 function rnd(x,f)
59 return fmt(type(x)=="number" and (x~x//1 and f or the.rnd) or "%s",x) end
60
61 function o(t, u,one)
62 one= function(k,v) return #t>0 and tostring(v) or fmt("%.3s %s",k,v) end
63 u={}; for k,v in pairs(t) do u[1+#u] = one(k,v) end
64 if #t==0 then sort(u) end
65 return (t.is or "").."["..table.concat(u," " ).."]" end
66
67 function coerce(x)
68 x = x:match("^%s*(-)%s*$")
69 if x=="true" then return true elseif x=="false" then return false end
70 return math.tointeger(x) or tonumber(x) or x end
71
72 function csv(src, cells)
73 t={}; for y in s:gmatch("(\\[\\+)" do t[1+#t]=coerce(y) end; return t end
74 src = io.input(src)
75 return function(x) x=io.read()
76 if x then return cells(x) else io.close(src) end end end
77
78 function workl(x, b4)
79 b4={}; for k,v in pairs(the) do b4[k]=v end
80 math.randomseed(the.seed)
81 if go[x] then print(x); go[x]() end
82 for k,v in pairs(b4) do the[k]=v end end
83
84 function work( t)
85 t={}; for k,_ in pairs(go) do push(t,k) end
86 for _,x in pairs(sort(t)) do workl(x) end end
87
88
89 local _id=0
90 function id() _id = _id+1; return _id end
91
92 function obj(name, t,new,str)
93 function new(kl,...)
94 local x=setmetatable({id=id(),kl}; kl.new(x,...); return x end
95 t = {__tostring=o, is=name or ""}; t.__index=t
96 return setmetatable(t, {__call=new}) end
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172

```

```

100 local Num=obj"Num"
101 function Num:new(at,tst)
102 self.at = at or 0
103 self.txt = tst or ""
104 self.n, self.mu, self.m2 = 0,0,0
105 self.w = self.txt:find"$" and -1 or 1
106 self.lo, self.hi = math.huge, -math.huge end
107
108 function Num:add(x, d)
109 if x ~="?" then
110 self.n = self.n + 1
111 self.lo = math.min(x, self.lo)
112 self.hi = math.max(x, self.hi)
113 d = x - self.mu
114 self.mu = self.mu + d/self.n
115 self.m2 = self.m2 + d*(x - self.mu) end
116 return x end
117
118 function Num:mid() return self.mu end
119 function Num:div() return (self.m2/(self.n - 1))^0.5 end
120
121 function Num:norm(x, lo,hi)
122 lo,hi = self.lo, self.hi
123 return x==?" and x or hi-lo < 1E-9 and 0 or (x - lo)/(hi - lo) end
124
125 function Num:dist(x,y)
126 if x==?" and y==?" then return 1 end
127 if x==?" then y = self:norm(y); x = y<.5 and 1 or 0
128 elseif y==?" then x = self:norm(x); y = x<.5 and 1 or 0
129 else x,y = self:norm(x), self:norm(y) end
130 return math.abs(x - y) end
131
132
133 local Sym=obj"Sym"
134 function Sym:new(at,tst)
135 self.at = at or 0
136 self.txt = tst or ""
137 self.n = 0
138 self.has, self.mode, self.most = {},nil,0 end
139
140 function Sym:add(x,inc)
141 if x ~="?" then
142 inc = inc or 1
143 self.n = self.n + inc
144 self.has[x] = inc + (self.has[x] or 0)
145 if self.has[x] > self.most then self.most,self.mode = self.has[x],x end end
146 return x end
147
148 function Sym:mid() return self.mode end
149 function Sym:div( e)
150 e=0; for _,v in pairs(t) do e=e-v/self.n*log(v/self.n,2) end; return e end
151
152 function Sym:dist(x,y) return x==?" and y==?" and 1 or x==y and 0 or 1 end
153
154
155 local Cols=obj"Cols"
156 function Cols:new(names, col)
157 self.names, self.all, self.x, self.y, self.klass = names, {}, {}, {}, nil
158 for at,tst in pairs(names) do
159 col = push(self.all, (txt:find"[A-Z]" and Num or Sym)(at,tst))
160 if not txt:find"$" then
161 if txt:find"$" then self.klass=col end
162 col.indep = not txt:find"|-$"
163 push(col.indep and self.x or self.y, col) end end end
164
165 function Cols:add(row)
166 for _,col in pairs(self.all) do col:add(row[col.at]) end
167 return row end
168
169
170 local Row=obj"Row"
171 function Row:new(t) self.cells = t end
172

```

```

172 -----
173 local Egs=obj"Egs"
174 function Egs:new() self.rows,self.cols = {}, nil end
175
176 function Egs:clone(rows, out)
177 out = Egs():add(self.cols.names)
178 for _,row in pairs(rows or {}) do out:add(row) end
179 return out end
180
181 function Egs:load(file)
182 for row in csv(file) do self:add(row) end; return self end
183
184 function Egs:add(t)
185 t = t.cells and t.cells or t
186 if self.cols
187 then push(self.rows, Row(self.cols:add(t)))
188 else self.cols=Cols(t) end
189 return self end
190
191 function Egs:better(row1,row2)
192 local s1, s2, n, e = 0, 0, #self.cols.y, math.exp(1)
193 for _,col in pairs(self.cols.y) do
194 local a = col:norm(row1.cells[col.at])
195 local b = col:norm(row2.cells[col.at])
196 s1 = s1 - e^(col.w * (a - b) / n)
197 s2 = s2 - e^(col.w * (b - a) / n) end
198 return s1 / n < s2 / n end
199
200 function Egs:betters(rows)
201 return sort(rows or self.rows, function(a,b) return self:better(a,b) end) end
202
203 function Egs:mid(cols)
204 return rnds(map(cols or self.cols.y, function(col) return col:mid() end)) end
205
206 function Egs:dist(row1,row2, d,n)
207 d = sum(self.cols.x, function(col)
208 return col:dist(row1.cells[col.at], row2.cells[col.at])^the.p end)
209 return (d / (#self.cols.x)) ^ (1/the.p) end
210
211 function Egs:around(row1, rows, around)
212 function around(row2) return (dist=self:dist(row1,row2),row=row2) end
213 return sort(map(rows or self.rows,around), lt"dist") end
214
215 function Egs:far(row, rows)
216 return per(self:around(row, rows or many(self.rows, the.some))).row end
217
218 function Egs:unsuper(n, recurse,known,rows,used,rest)
219 function known(row) used[row.id]=true; return row end
220 function recurse(rows,some,x, y,tmp)
221 if #rows <= 20 then
222 oo(self:clone(rows):mid())
223 else
224 x = known( x or self:far(any(some),some))
225 y = known( self:far(x,some))
226 if self:better(y, x) then x,y = y,x end
227 tmp = {}
228 for _,row in pairs(rows) do
229 push(self:dist(row,x) < self:dist(row,y) and tmp or rest, row) end
230 recurse(tmp, many(tmp,n), x) end
231 end -----
232 used, rest = {}, {}
233 recurse(self.rows, many(self.rows,n)) end
234
235 -----
236 fails,go,no = 0,{},{}
237 function ok(test,msg)
238 print("", test and "PASS"or "FAIL ", msg or "")
239 if not test then
240 fails= fails+1
241 if the.dump then assert(test,msg) end end end
242
243 function go.many()
244 oo(many({10,20,30,40,50,60,70,80,90,100},3)) end
245
246 function go.unsuper( eg)
247 eg = Egs():load(the.file)
248 for i=1,10 do eg:unsuper(64) end end
249
250 function go.egl( eg)
251 eg = Egs():load(the.file)
252 print(eg.cols.y[1]) end
253
254 function go.dist( eg,row2,t)
255 eg = Egs():load(the.file)
256 t={}; for i=1,20 do
257 row2= any(eg.rows)
258 push(t, {dist=eg:dist(eg.rows[1],row2), row = row2}) end
259 oo(eg.rows[1])
260 for _,two in pairs(sort(t,lt"dist")) do oo(two.row.cells) end end
261
262 function go.mids( eg,hi,lo,out)
263 eg = Egs():load(the.file)
264 oo(map(eg.cols.y,function(col) return col.txt end))
265 oo(eg:mid())
266 lo,hi = eg:clone(), eg:clone()
267 for i,row in pairs(eg.rows) do
268 if i < 20 then lo:add(row) end
269 if i > #eg.rows - 20 then hi:add(row) end end
270 oo(lo:mid())
271 oo(hi:mid()) end
272
273 -----
274 help:gsusb("\n ([-|([^\%s|+)]|[%s|+(-|^\%s|+)]^\n)%s([^\%s|+)",
275 function(long,key,short,x)
276 for n,flag in ipairs(arg) do
277 if flag==short or flag==long then
278 x = x=="false" and "true" or x=="true" and "false" or arg[n+1] end end
279 the[key] = coerce(x) end)
280
281 if the.help then print(help) end
282 if the.todo=="all" then work1() else work1(the.todo) end
283 for k,v in pairs(_ENV) do if not b4[k] then print("? ",k,type(v)) end end
284 os.exit(fails)

```