

```

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
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

```



```

local b4={}; for k,_ in pairs(_ENV) do b4[k]=k end
local the, help = {}, {}

lua bnbad.lua [OPTIONS]
(c) 2022, Tim Menzies, opensource.org/licenses/BSD-2-Clause

OPTIONS:
-c cohen                = .35
-f far                  = .9
-k keep                 = 256
-m minitems             = .5
-p euclidean coefficient = 3

OPTIONS, other:
-d dump                 = false
-f file                 = ../etc/data/auto93.csv
-h help                 = false
-rnd                    = %5.2f
-s seed                 = 10019
-t todo                 = nothing
]]

local any, bestSpan, bins, bins1, bootstrap, firsts, fmt, last
local many, map, new, o, obj, oo, per, push, quintiles, r, rnd, rnds, scottKnot
local selects, settings, slots, smallfx, sort, sum, thing, things, xplains

-- Copyright 2022 Tim Menzies
--
-- Redistribution and use in source and binary forms, with or without
-- modification, are permitted provided that the following conditions
-- are met:
--
-- 1. Redistributions of source code must retain the above copyright
-- notice, this list of conditions and the following disclaimer.
--
-- 2. Redistributions in binary form must reproduce the above copyright
-- notice, this list of conditions and the following disclaimer in the
-- documentation and/or other materials provided with the distribution.
--
-- THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
-- "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
-- LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
-- FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
-- COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
-- INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,
-- BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
-- LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
-- CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
-- LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
-- ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
-- POSSIBILITY OF SUCH DAMAGE.

```

```

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
173
174
175
176

```

CLASSES

```

new = setmetatable
function obj(s, t)
  t={__toString=o, __is=s or ""}; t.__index=t
  return new(t, {__call=function(_,...) return t.new(_,...) end}) end

local Num, Sym, Egs = obj"Num", obj"Sym", obj"Egs"

--
--
--
function Sym:new(at,name)
  return new({at=at, name=name, most=0,n=0,all={}}, Sym) end

function Num:new(at,name)
  return new({at=at, name=name, __all={}, w=(name or ""):find"-"$ and ~1 or 1,
    n=0, sd=0, mu=0, m2=0, lo=math.huge, hi=-math.huge}, Num) end

function Egs:new(names, i,col)
  i = new({all={}, cols={names=names, all={}, x={}, y={}}, Egs)
  for at,name in pairs(names) do
    col = (name:find"^[A-Z]~" and Num or Sym)(at,name)
    push(i.cols.all, col)
    if not name:find"$" then
      if name:find"$" then i.cols.class = col end
      push(name:find"[~+!]"$ and i.cols.y or i.cols.x, col) end end
  return i end

--
--
--
function Sym.copy(i) return Sym(i.at, i.name) end

function Num.copy(i) return Num(i.at, i.name) end

function Egs.copy(i,all, j)
  j = Egs(i.cols.name)
  for _,row in pairs(rows or {}) do i:add(row) end
  return j end

--
--
--
function Egs.add(i,row)
  i.all[1 + #i.all] = row
  for at,col in pairs(i.cols) do col:add(row[col.at]) end end

function Sym.add(i,x,inc)
  if x ~ "?" then
    inc = inc or 1
    i.n = i.n+inc
    i.all[x] = inc + (i.all[x] or 0)
    if i.all[x] > i.most then i.most, i.mode = i.all[x], x end end end

function Sym.sub(i,x,inc)
  if x ~ "?" then
    inc = inc or 1
    i.n = i.n - inc
    i.all[x] = i.all[x] - inc end end

function Num.add(i,x,_, d,a)
  if x ~ "?" then
    i.n = i.n + 1
    d = x - i.mu
    i.mu = i.mu + d/i.n
    i.m2 = i.m2 + d*(x - i.mu)
    i.sd = (i.m2<0 or i.n<2) and 0 or ((i.m2/(i.n - 1))^0.5)
    i.lo = math.min(x, i.lo)
    i.hi = math.max(x, i.hi)
    a = i.__all
    if #a < the.keep then i.ok=false; push(a,x)
    elseif r() < the.keep/i.n then i.ok=false; a[r(#a)]=x end end end

function Num.sub(i,x,_, d)
  if x ~ "?" then
    i.n = i.n - 1
    d = x - i.mu
    i.mu = i.mu - d/i.n
    i.m2 = i.m2 - d*(x - i.mu)
    i.sd = (i.m2<0 or i.n<2) and 0 or ((i.m2/(i.n - 1))^0.5) end end

--
--
--
function Num.sorted(i)
  if not i.ok then table.sort(i.__all); i.ok=true end
  return i.__all end

function Num.mid(i) return i.mu end
function Sym.mid(i) return i.mode end

function Num.div(i) return i.sd end
function Sym.div(i, e)
  e=0
  for _,n in pairs(i.all) do
    if n > 0 then e = n/i.n * math.log(n/i.n,2) end end
  return -e end

function Num.norm(i,x)
  return i.hi - i.lo < 1E-32 and 0 or (x - i.lo)/(i.hi - i.lo) end

```

```

177 --- c | l | a | t | a |
178 ---
179 function Num.dist(i,a,b)
180 if a=="?" and b=="?" then return 1 end
181 if a=="?" then b=i:norm(b); a=b<.5 and 1 or 0
182 elseif b=="?" then a=i:norm(a); b=a<.5 and 1 or 0
183 else a,b = i:norm(a), i:norm(b) end
184 return math.abs(a - b) end
185
186
187 function Sym.dist(i,a,b)
188 return a=="?" and b=="?" and 1 or a==b and 0 or 1 end
189
190 function Egs.dist(i,row1,row2,d)
191 d = sum(i.cols.x, function(c) return c:dist(row1[c.at], row2[c.at])^the.p end)
192 return (d/#i.cols.x)^(1/the.p) end
193
194 function Egs.dists(i,r1,rows)
195 return sort(map(rows,function(s) return{i:dist(r1,r2),r2} end),firsts) end
196
197 function Egs.half(i, rows)
198 local project,far,some,left,right,c,lefts,rights
199 far = function(r,t) return per(i:dists(r,t), the.far)[2] end
200 project= function(r1, a,b)
201 a,b = i:dist(left,r1), i:dist(right,r1)
202 return {(a^2 + c^2 - b^2)/(2*c), r1} end
203 some = many(rows, the.some)
204 left = i:far(any(some), some)
205 right = i:far(left, some)
206 c = i:dist(left, right)
207 lefts,rights = i:copy(), i:copy()
208 for n, projection in pairs(sort(map(rows,project),firsts)) do
209 if n==#rows//2 then mid=row end
210 (n <= #rows//2 and lefts or rights):add( projection[2] ) end
211 return lefts, rights, left, right, mid, c end
212
213 --- d | i | s | c | r | e | t | i | z | e
214 ---
215 function Num.spans(i,j, cuts)
216 local xys,all = {}, Num
217 for _,n in pairs(i..all) do all:add(n); push(xys,{x=n,y="left"}) end
218 for _,n in pairs(j..all) do all:add(n); push(xys,{x=n,y="right"}) end
219 return bins(i,cuts,
220 bins1(sort(xys,first), (#xys)^the.minItems,all.sd*the.cohen,Sym,{})) end
221
222 function bins1(col, old,new)
223 if #new1 then
224 new[1].lo = -math.huge
225 new[#new].hi = math.huge
226 for _,cut in pairs(new) do cut.col= col; push(old,cut) end end end
227
228
229 function bins1(xys, minItems, cohen, yclass, cuts, b4)
230 local lhs, rhs, b4, cut, div, xpect = yclass(), yclass(), b4 or xys[1].x
231 function xpect(i,j) return (i.n*i:div() + j.n*j:div()) / (i.n + j.n) end
232 for _,xy in pairs(xys) do rhs:add(xy.y) end
233 div = rhs:div()
234 for j,xy in pairs(xys) do
235 lhs:add(xy.y)
236 rhs:sub(xy.y)
237 if lhs.n >= minItems and rhs.n >= minItems then
238 if xy.x ~= xys[j+1].x then
239 if xy.x - xys[1].x >= cohen and xys[#xys].x - xy.x >= cohen then
240 if xpect(lhs,rhs) < div then
241 cut, div = j, xpect(lhs,rhs) end end end end end
242
243 if cut
244 then local l,r = {},{}
245 for n,xy in pairs(xys) do push(n<=cut and l or r, xy) end
246 bins1(l, minItems, cohen, yclass, cuts, b4)
247 bins1(r, minItems, cohen, yclass, cuts, xys[cut].x)
248 else push(cuts, {lo=b4, hi=xys[#xys].x, n=#xys, div=div}) end end
249
250 ---
251 --- >|plain
252 ---
253 local xplain,xplains,selects,spanShow
254 function Egs.xplain(i,rows)
255 local stop,here,left,right,lefts0,rights0,lefts1,rights1
256 rows = rows or i.all
257 here = {all=rows}
258 stop = (#i.all)^the.minItems
259 if #rows >= 2*stop then
260 lefts0, rights0, here.left, here.right, here.mid, here.c = half(i, rows)
261 if #lefts0.all < #rows then
262 cuts = {}
263 for j,col in pairs(lefts0.col.x) do col:spans(rights0.col.x[j],cuts) end
264 lefts1,rights1 = {},{}
265 for _,row in pairs(rows) do
266 push(selects(here.selector, row) and lefts1 or rights1, row) end
267 if #lefts1 > stop then here.lefts = xplain(i,lefts1) end
268 if #rights1 > stop then here.rights = xplain(i,rights1) end end end
269 return here end
270
271 function bestSpan(spans)
272 local divs,ns,n,div,stats,dist2heaven = Num(), Num()
273 function dist2heaven(s) return {(1 - n(s))^2 + (0 - div(s))^2^.5,s} end
274 function div(s) return divs:norm( s.all:div() ) end
275 function n(s) return ns:norm( s.all.n ) end
276 for _,s in pairs(spans) do
277 add(divs, s.all:div())
278 add(ns, s.all.n) end
279 return sort(map(spans, dist2heaven), firsts)[1][2] end
280
281 function selects(span,row, lo,hi,at,x)
282 lo, hi, at = span.lo, span.hi, span.col.at
283 x = row[at]
284 if x=="?" then return true end
285 if lo==hi then return x==lo else return lo <= x and x < hi end end
286
287 function xplains(i,format,t,pre,how, sel,front)
288 pre, how = pre or "", how or ""
289 if t then
290 pre=pre or ""
291 front = fmt("%s%s%s",pre,how, #t.all, t.c and rnd(t.c) or "")
292 if t.lefts and t.rights then print(fmt("%-35s",front)) else
293 print(fmt("%-35s",front, o(rnds(mids(i,t.all),format))))
294 end
295 sel = t.selector
296 xplains(i,format,t.lefts, "... pre, spanShow(sel,":")")
297 xplains(i,format,t.rights, "... pre, spanShow(sel,true) ..:") end end

```

```

298 --- s | t | a | r | t |
299 ---
300 function quintiles(ts,width, nums,out,all,n,m)
301 width=width or 32
302 nums=Num(); for _,t in pairs(ts) do
303 all,out = nums.all, {}
304 for _,x in pairs(sort(t)) do add(nums,x) end end
305
306 local s, where = {}
307 where = function(n) return (width*nums:norm(n))/1 end
308 for j = 1, width do s[j]=" " end
309 for j = where(per(t,.1)), where(per(t,.3)) do s[j]="-" end
310 for j = where(per(t,.7)), where(per(t,.9)) do s[j]="-" end
311 s[where(per(t,.5))] = "|"
312 push(out,{display=table.concat(s),
313 data = t,
314 pers = map({.1,.3,.5,.7,.9},
315 function(p) return rnd(per(t,p)) end)}) end
316
317 return out end
318
319 function smallfx(xs,ys, x,y,lt,gt,n)
320 lt,gt,n = 0,0,0
321 if #ys > #xs then xs,ys=ys,xs end
322 for _,x in pairs(xs) do
323 for j=1, math.min(64,#ys) do
324 y = any(ys)
325 if y<x then lt=lt+1 end
326 if y>x then gt=gt+1 end
327 n = n+1 end end
328 return math.abs(gt - lt) / n <= the.cliffs end
329
330 function bootstrap(y0,z0)
331 local x, y, z, b4, yhat, zhat, bigger
332 local function obs(a,b, c)
333 c = math.abs(a.mu - b.mu)
334 return (a.sd + b.sd) == 0 and c or c/((x.sd^2/x.n + y.sd^2/y.n)^.5) end
335 local function adds(t, num)
336 num = num or Num(); map(t, function(x) add(num,x) end); return num end
337 y,z = adds(y0), adds(z0)
338 x = adds(y0, adds(z0))
339 b4 = obs(y,z)
340 yhat = map(y.all, function(y1) return y1 - y.mu + x.mu end)
341 zhat = map(z.all, function(z1) return z1 - z.mu + x.mu end)
342 bigger = 0
343 for j=1,the.boot do
344 if obs( adds(many(yhat,#yhat)), adds(many(zhat,#zhat))) > b4
345 then bigger = bigger + 1/the.boot end
346 return bigger >= the.conf end
347
348 --- xxx mid has to be per and
349 -- XXXX implement same
350 -- XXXX need tests for stats
351 function scottKnot(nums, all,cohen)
352 local mid = function(z) return z.some:mid()
353 end
354 local function summary(i,j, out)
355 out = copy(nums[i])
356 for k = i+1, j do out = out:merge(nums[k]) end
357 return out
358 end
359 local function div(lo,hi,rank,b4, cut,best,l,1l,r,r1,now)
360 best = 0
361 for j = lo,hi do
362 if j < hi then
363 l = summary(lo, j)
364 r = summary(j+1, hi)
365 now = (l.n*(mid(l) - mid(b4))^2 + r.n*(mid(r) - mid(b4))^2)
366 / (l.n + r.n)
367 if now > best then
368 if math.abs(mid(l) - mid(r)) >= cohen then
369 cut, best, 1l, r1 = j, now, copy(l), copy(r)
370 end end end end
371 if cut and not 1l:same(r1,the) then
372 rank = div(lo, cut, rank, 1l) + 1
373 rank = div(cut+1, hi, rank, r1)
374 else
375 for i = lo,hi do nums[i].rank = rank end end
376 return rank
377 end
378 table.sort(nums, function(x,y) return mid(x) < mid(y) end)
379 all = summary(1,#nums)
380 cohen = all.sd * the.cohen
381 div(1, #nums, 1, all)
382 return nums end

```

