

```

1  -----
2  ---
3  ---
4  ---
5  ---
6  ---
7  ---
8  ---
9  ---
10 ---
11 ---
12 ---
13 ---
14 ---
15 ---
16 ---
17 ---
18 ---
19 ---
20 ---
21 ---
22 lua brknbad.lua [OPTIONS]
23 (c) 2022, Tim Menzies, BSD-2-Clause
24 Divide things. Show deltas between things.
25
26 OPTIONS:
27 -cohen      -c cohen              = .35
28 -far        -f how far to seek poles = .9
29 -keep       -k items to keep      = 256
30 -minitems   -m min items in a range = .5
31 -p          -p euclidean coefficient = 3
32
33 OPTIONS, other:
34 -dump       -d stackdump on error  = false
35 -file       -f data file           = ../etc/data/auto93.csv
36 -help       -h show help           = false
37 -rnd        -r round numbers       = %5.2f
38 -seed       -s random number seed = 10019
39 -todo       -t start-up action     = nothing
40
41 ]]
42
43 local any, bestSpan, bins, bins1, bootstrap, csv2egs, firsts, fmt, ish, last
44 local many, map, new, o, obj, oo, per, push, quintiles, r, rnd, rnds, scottKnot
45 local selects, settings, slots, smallfx, sort, sum, thing, things, xplains
46 local Num, Sym, Egs
47
48 -- Copyright 2022 Tim Menzies
49
50 -- Redistribution and use in source and binary forms, with or without
51 -- modification, are permitted provided that the following conditions
52 -- are met:
53 -- 1. Redistributions of source code must retain the above copyright
54 -- notice, this list of conditions and the following disclaimer.
55 --
56 -- 2. Redistributions in binary form must reproduce the above copyright
57 -- notice, this list of conditions and the following disclaimer in the
58 -- documentation and/or other materials provided with the distribution.
59 --
60 -- THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
61 -- "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
62 -- LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
63 -- FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
64 -- COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
65 -- INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,
66 -- BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
67 -- LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
68 -- CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
69 -- LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
70 -- ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
71 -- POSSIBILITY OF SUCH DAMAGE.

```

```

72  -----
73  ---
74  ---
75  ---
76  ---
77  ---
78  ---
79  ---
80 r=math.random
81 function ish(x,y,z) return math.abs(y -x ) < z end
82
83 ---
84 ---
85 ---
86 ---
87 function any(a)      return a[ math.random(#a) ] end
88 function firsts(a,b) return a[1] < b[1] end
89 function last(a)     return a[ #a ] end
90 function many(a,n, u) u={}; for j=1,n do push(u,any(a)) end; return u end
91 function map(t,f, u) u={};for _,v in pairs(t) do push(u,f(v)) end;return u end
92 function per(a,p)    return a[ (p*#a)//1 ] end
93 function push(t,x)   t[1 + #t] = x; return x end
94 function sort(t,f)   table.sort(t,f); return t end
95 function sum(t,f, n) f = f or function(x) return x end
96                     n=0; for _,v in pairs(t) do n = n + f(v) end; return n end
97
98 ---
99 ---
100 ---
101 ---
102 ---
103 ---
104 function thing(x)
105   x = x:match("^%s*(-)%s*$")
106   if x=="true" then return true elseif x=="false" then return false end
107   return tonumber(x) or x end
108
109 function things(file, x)
110   local function cells(x, t)
111     t={}; for v in x:gmatch("[^,]+") do push(t, thing(v)) end; return t end
112   file = io.input(file)
113   return function()
114     x=io.read(); if x then return cells(x) else io.close(file) end end end
115
116 function csv2egs(file, eggs)
117   for row in things(the.file) do
118     if eggs then eggs:add(row) else eggs=Egs(row) end end
119   return eggs end
120
121 ---
122 ---
123 ---
124 ---
125 fmt = string.format
126
127 function oo(t) print(o(t)) end
128
129 function o(t, seen, u)
130   if type(t)~="table" then return tostring(t) end
131   seen = seen or {}
132   if seen[t] then return "..." end
133   seen[t] = t
134   local function show1(x) return o(x, seen) end
135   local function show2(k) return fmt("%s %s", k, o(t[k],seen)) end
136   u = #t>0 and map(t,show1) or map(slots(t),show2)
137   return (t._is or "")..["%.table.concat(u, ",").." ]" end
138
139 function slots(t, u)
140   u={};for k,v in pairs(t) do if tostring(k):sub(1,1)~="_" then push(u,k) end end
141   return sort(u) end
142
143 function rnds(t,f) return map(t, function(x) return rnd(x,f) end) end
144 function rnd(x,f)
145   return fmt(type(x)=="number" and (x~=x//1 and f or the.rnd) or "%s",x) end
146
147 ---
148 ---
149 ---
150 ---
151 function settings(txt, d)
152   d={ }
153   txt:gsub("\n ([-]([^%$+))[%$]+(-[%$+][^%$+]*%$([^%$+])",
154     function(long,key,short,x)
155       for n,flag in ipairs(arg) do
156         if flag==short or flag==long then
157           x = x=="false" and true or x=="true" and "false" or arg[n+1] end end
158       d[key] = x==true and true or thing(x) end
159   if d.help then print(txt) end
160   return d end
161
162 ---
163 ---
164 ---
165 ---
166 local go, ok = {fails=0}
167 function ok(test,msg)
168   print(test and " PASS:" or " FAIL:" ,msg or "")
169   if not test then
170     go.fails = go.fails+1
171     if the.dump then assert(test,msg) end end end
172
173 function go.main(todo,seed)
174   for k,one in pairs(todo=="all" and slots(go) or {todo}) do
175     if k ~= "main" and type(go[one]) == "function" then
176       math.randomseed(seed)
177       print(fmt(":%s",one))
178       go[one]() end end
179   for k,v in pairs(_ENV) do if not b4[k] then print("?",k,type(v)) end end end
180
181 ---
182 ---
183 ---
184 new = setmetatable
185 function obj(s, t)
186   t={__tostring=o,_is=s or ""}; t._index=t
187   return new(t, {__call=function(_,...) return t.new(_,...) end}) end

```

```

188 -----
189 --- CLASSES
190
191
192
193 Num, Sym, Egs = obj"Num", obj"Sym", obj"Egs"
194
195 --- create
196
197 function Sym:new(at,name)
198     return new({at=at, name=name, most=0,n=0,all={}}, Sym) end
199
200
201 function Num:new(at,name)
202     return new({at=at, name=name, _all={}, w=(name or ""):find"$" and -1 or 1,
203               n=0, sd=0, mu=0, m2=0, lo=math.huge, hi=-math.huge), Num) end
204
205
206 function Egs:new(names, i,col)
207     i = new({_all={}, cols={names=names, all={}, x={}, y={}}, Egs)
208     for at,name in pairs(names) do
209         col = push(i.cols.all, (name:find"[A-Z]" and Num or Sym) (at,name) )
210         if not name:find"$" then
211             if name:find"$" then i.cols.class = col end
212             push(name:find"[+!$]" and i.cols.x or i.cols.y, col) end end
213     return i end
214
215 --- copy
216
217 function Sym.copy(i) return Sym(i.at, i.name) end
218
219 function Num.copy(i) return Num(i.at, i.name) end
220
221
222 function Egs.copy(i,all, j)
223     j = Egs(i.cols.name)
224     for _,row in pairs(rows or {}) do i:add(row) end
225     return j end
226
227
228 --- update
229
230
231 function Egs.add(i,row)
232     push(i._all, row)
233     for at,col in pairs(i.cols.all) do col:add(row[col.at]) end end
234
235
236 function Sym.add(i,x,inc)
237     if x == "?" then
238         inc = inc or 1
239         i.n = i.n+inc
240         i.all[x] = inc + (i.all[x] or 0)
241         if i.all[x] > i.most then i.most, i.mode = i.all[x], x end end end
242
243 function Sym.sub(i,x,inc)
244     if x == "?" then
245         inc = inc or 1
246         i.n = i.n - inc
247         i.all[x] = i.all[x] - inc end end
248
249 function Num.add(i,x,_, d,a)
250     if x == "?" then
251         i.n = i.n + 1
252         d = x - i.mu
253         i.mu = i.mu + d/i.n
254         i.m2 = i.m2 + d*(x - i.mu)
255         i.sd = (i.m2<0 or i.n<2) and 0 or ((i.m2/(i.n - 1))^0.5)
256         i.lo = math.min(x, i.lo)
257         i.hi = math.max(x, i.hi)
258         a = i._all
259         if #a < the.keep then i.ok=false; push(a,x)
260         elseif r() < the.keep/i.n then i.ok=false; a[r(#a)]=x end end end
261
262 function Num.sub(i,x,_, d)
263     if x == "?" then
264         i.n = i.n - 1
265         d = x - i.mu
266         i.mu = i.mu - d/i.n
267         i.m2 = i.m2 - d*(x - i.mu)
268         i.sd = (i.m2<0 or i.n<2) and 0 or ((i.m2/(i.n - 1))^0.5) end end
269
270 --- display
271
272
273 function Num.all(i)
274     if not i.ok then table.sort(i._all); i.ok=true end
275     return i._all end
276
277 function Num.mid(i) return i.mu end
278 function Sym.mid(i) return i.mode end
279
280 function Num.div(i) return i.sd end
281 function Sym.div(i, e)
282     e=0
283     for _,n in pairs(i.all) do
284         if n > 0 then e = e + n/i.n * math.log(n/i.n,2) end end
285     return -e end
286
287
288 function Num.norm(i,x)
289     return i.hi - i.lo < 1E-32 and 0 or (x - i.lo)/(i.hi - i.lo) end
290
291
292 --- class
293
294
295 function Num.dist(i,a,b)
296     if a=="?" and b=="?" then return 1 end
297     if a=="?" then b=i:norm(b); a=b<.5 and 1 or 0
298     elseif b=="?" then a=i:norm(a); b=a<.5 and 1 or 0
299     else a,b = i:norm(a), i:norm(b) end
300     return math.abs(a - b) end
301
302
303 function Sym.dist(i,a,b)
304     return a=="?" and b=="?" and 1 or a==b and 0 or 1 end
305
306
307 function Egs.dist(i,row1,row2, d)
308     d = sum(i.cols.x, function(c) return c:dist(row1[c.at], row2[c.at])^the.p end)
309     return (d/#i.cols.x)^(1/the.p) end
310
311
312 function Egs.dists(i,r1,rows)
313     return sort(map(rows,function(s) return {i:dist(r1,r2),r2} end),firsts) end
314
315
316 function Egs.half(i, rows)
317     local project,far,some,left,right,c,lefts,rights
318     far = function(r,t) return per(i:dists(r,t), the.far) [2] end
319     project = function(r1, a,b)
320         a,b = i:dist(left,r1), i:dist(right,r1)
321         return ((a^2 + c^2 - b^2)/(2*c), r1) end
322     some = many(rows, the.some)
323     left = i:far(any(some), some)
324     right = i:far(left, some)
325     c = i:dist(left, right)
326     lefts,rights = i:copy(), i:copy()
327     for n,projection in pairs(sort(map(rows,project),firsts)) do
328         if n==#rows//2 then mid=row end
329         (n <= #rows//2 and lefts or rights):add( projection[2] ) end
330     return lefts, rights, left, right, mid, c end
331
332 --- discretize
333
334
335 function Num.spans(i,j, cuts)
336     local xys,all = {}, Num
337     for _,n in pairs(i._all) do all:add(n); push(xys,{x=n,y="left"}) end
338     for _,n in pairs(j._all) do all:add(n); push(xys,{x=n,y="right"}) end
339     return bins(i,cuts,
340               bins1(sort(xys,first), (#xys)^the.minItems,all.sd*the.cohen,Sym,{})) end
341
342
343 function bins1(col, old,new)
344     if #new1 then
345         new[1].lo = -math.huge
346         new[#new].hi = math.huge
347         for _,cut in pairs(new) do cut.col= col; push(old,cut) end end end
348
349 function bins1(xys, minItems, cohen, yclass, cuts, b4)
350     local lhs, rhs, b4, cut, div, xpect = yclass(), yclass(), b4 or xys[1].x
351     function xpect(i,j) return (i.n*i:div() + j.n*j:div()) / (i.n + j.n) end
352     for _,xy in pairs(xys) do rhs:add(xy.y) end
353     div = rhs:div()
354     for j,xy in pairs(xys) do
355         lhs:add(xy.y)
356         rhs:sub(xy.y)
357         if lhs.n >= minItems and rhs.n >= minItems then
358             if xy.x == xys[j+1].x then
359                 if xy.x - xys[1].x >= cohen and xys[#xys].x - xy.x >= cohen then
360                     if xpect(lhs,rhs) < div then
361                         cut, div = j, xpect(lhs,rhs) end end end end end
362             if cut
363             then local l,r = {},{}
364                 for n,xy in pairs(xys) do push(n<=cut and l or r, xy) end
365                 bins1(l, minItems, cohen, yclass, cuts, b4)
366                 bins1(r, minItems, cohen, yclass, cuts, xys[cut].x)
367             else push(cuts, {lo=b4, hi=xys[#xys].x, n=#xys, div=div}) end end
368         end
369     end
370
371 --- explain
372
373
374 local xplain,xplains,selects,spanShow
375 function Egs.xplain(i,rows)
376     local stop,here,left,right,lefts0,rights0,lefts1,rights1
377     rows = rows or i._all
378     here = {all=rows}
379     stop = (#i._all)^the.minItems
380     if #rows >= 2*stop then
381         lefts0, rights0, here.left, here.right, here.mid, here.c = half(i, rows)
382         if #lefts0._all < #rows then
383             cuts = {}
384             for j,col in pairs(lefts0.col.x[j],cuts) end
385             lefts1,rights1 = {},{}
386             for _,row in pairs(rows) do
387                 push(selects(here.selector, row) and lefts1 or rights1, row) end
388             if #lefts1 > stop then here.lefts = xplain(i,lefts1) end
389             if #rights1 > stop then here.rights = xplain(i,rights1) end end end
390     return here end
391
392
393 function xbestSpan(spans)
394     local divs,ns,n,div,stats,dist2heaven = Num(), Num()
395     function dist2heaven(s) return {(1 - n(s))^2 + (0 - div(s))^2^.5,s} end
396     function div(s) return divs:norm( s.all:div() ) end
397     function n(s) return ns:norm( s.all.n ) end
398     for _,s in pairs(spans) do
399         add(divs, s.all:div())
400         add(ns, s.all.n) end
401     return sort(map(spans, dist2heaven), firsts) [1] [2] end
402
403
404 function selects(span,row, lo,hi,at,x)
405     lo, hi, at = span.lo, span.hi, span.col.at
406     x = row[at]
407     if x=="?" then return true end
408     if lo==hi then return x==lo else return lo <= x and x < hi end end
409
410 function xplains(i,format,t,pre,how, sel,front)
411     pre, how = pre or "", how or ""
412     if t then
413         pre=pre or ""
414         front = fmt("%s%s%s",pre,how, #t.all, t.c and rnd(t.c) or "")
415         if t.lefts and t.rights then print(fmt("%-35s",front)) else
416             print(fmt("%-35s",front, o(rnds(mids(i,t.all),format))))
417         end
418         sel = t.selector
419         xplains(i,format,t.lefts, " |... pre, spanShow(sel)..:")
420         xplains(i,format,t.rights, " |... pre, spanShow(sel,true) ..:") end end

```

```

410 ---
411 ---
412
413 function quintiles(ts,width,  nums,out,all,n,m)
414 width=width or 32
415 nums=Num(); for _,t in pairs(ts) do
416     for _,x in pairs(sort(t)) do add(nums,x) end end
417 all,out = nums.all, {}
418 for _,t in pairs(ts) do
419     local s, where = {}
420     where = function(n) return (width*nums:norm(n))/1 end
421     for j = 1, width do s[j]="" end
422     for j = where(per(t,.1)), where(per(t,.3)) do s[j]="-" end
423     for j = where(per(t,.7)), where(per(t,.9)) do s[j]="-" end
424     s[where(per(t,.5))] = "|"
425     push(out,{display=table.concat(s),
426         data = t,
427         pers = map({.1,.3,.5,.7,.9},
428             function(p) return rnd(per(t,p))end)}) end
429
430 return out end
431
432 function smallfx(xs,ys,      x,y,lt,gt,n)
433 lt,gt,n = 0,0,0
434 if #ys > #xs then xs,ys=ys,xs end
435 for _,x in pairs(xs) do
436     for j=1, math.min(64,#ys) do
437         y = any(ys)
438         if y<x then lt=lt+1 end
439         if y>x then gt=gt+1 end
440         n = n+1 end end
441 return math.abs(gt - lt) / n <= the.cliffs end
442
443 function bootstrap(y0,z0)
444 local x, y, z, b4, yhat, zhat, bigger
445 local function obs(a,b, c)
446     c = math.abs(a.mu - b.mu)
447     return (a.sd + b.sd) == 0 and c or c/((x.sd^2/x.n + y.sd^2/y.n)^.5) end
448 local function adds(t, num)
449     num = num or Num(); map(t, function(x) add(num,x) end); return num end
450 y,z = adds(y0), adds(z0)
451 x = adds(y0, adds(z0))
452 b4 = obs(y,z)
453 yhat = map(y._all, function(y1) return y1 - y.mu + x.mu end)
454 zhat = map(z._all, function(z1) return z1 - z.mu + x.mu end)
455 bigger = 0
456 for j=1,the.boot do
457     if obs( adds(many(yhat,#yhat)), adds(many(zhat,#zhat))) > b4
458     then bigger = bigger + 1/the.boot end end
459 return bigger >= the.conf end
460
461 --- xxx mid has to be per and
462 -- XXX implement same
463 -- XXX need tests for stats
464 function scottKnot(nums,      all,cohen)
465 local mid = function(z) return z.some:mid()
466 end
467 local function summary(i,j,      out)
468     out = copy(nums[i])
469     for k = i+1, j do out = out:merge(nums[k]) end
470     return out
471 end
472 local function div(lo,hi,rank,b4,      cut,best,l,ll,r,r1,now)
473     best = 0
474     for j = lo,hi do
475         if j < hi then
476             l = summary(lo, j)
477             r = summary(j+1, hi)
478             now = (l.n*(mid(l) - mid(b4))^2 + r.n*(mid(r) - mid(b4))^2) / (l.n + r.n)
479             if now > best then
480                 if math.abs(mid(l) - mid(r)) >= cohen then
481                     cut, best, ll, r1 = j, now, copy(l), copy(r)
482                 end end end
483             if cut and not ll:same(r1,the) then
484                 rank = div(lo,      cut, rank, ll) + 1
485                 rank = div(cut+1, hi, rank, r1)
486             else
487                 for i = lo,hi do nums[i].rank = rank end end
488             return rank
489         end
490     end
491 table.sort(nums, function(x,y) return mid(x) < mid(y) end)
492 all = summary(1,#nums)
493 cohen = all.sd * the.cohen
494 div(1, #nums, 1, all)
495 return nums end

```

```

496 -----
497 ---
498 ---
499 ---
500 function go.last()
501     ok( 30 == last({10,20,30}, "lasts") end
502
503 function go.per( t)
504     t={};for i=1,100 do push(t,i*1000) end
505     ok(70000 == per(t,.7), "per") end
506
507 function go.many( t)
508     t={};for i=1,100 do push(t,i) end; many(t,10) end
509
510 function go.sum( t)
511     t={};for i=1,100 do push(t,i) end; ok(5050==sum(t),"sum")end
512
513 function go.sample( m,n)
514     m,n = 10^5,Num(); for i=1,m do n:add(i) end
515     for j=.1,.9,.1 do
516         print(j,per(n:all(),j),ish(per(n:all(),j),m*j,m*0.05)) end end
517
518 function go.sym( s)
519     s=Sym(); map({1,1,1,2,2,3}, function(x) s:add(x) end)
520     ok(ish(s:div(),1.378, 0.001), "cnt") end
521
522 function go.num( n)
523     n=Num(); map({10, 12, 23, 23, 16, 23, 21, 16}, function(x) n:add(x) end)
524     print(n:div())
525     ok(ish(n:div(),5.895,0.001), "div") end
526
527 function go.nums( num,t,b4)
528     b4,t,num={}, {},Num()
529     for j=1,1000 do push(t,100*r() *j) end
530     for j=1,#t do
531         num:add(t[j])
532         if j%100==0 then b4[j] = fmt("%.5f",num:div()) end end
533     for j=#t,1,-1 do
534         if j%100==0 then ok(b4[j] == fmt("%.5f",num:div()),"div"..j) end
535         num:sub(t[j]) end end
536
537 function go.syms( t,b4,s,sym)
538     b4,t,sym, s={}, {},Sym(), "I have gone to seek a great perhaps."
539     t={}; for j=1,20 do s:gsup('.',function(x) t[#t+1]=x end) end
540     for j=1,#t do
541         sym:add(t[j])
542         if j%100==0 then b4[j] = fmt("%.5f",sym:div()) end end
543     for j=#t,1,-1 do
544         if j%100==0 then ok(b4[j] == fmt("%.5f",sym:div()),"div"..j) end
545         sym:sub(t[j]) end
546     end
547
548 function go.loader( num)
549     for row in things(the.file) do
550         if num then num:add(row[1]) else num=Num() end end
551     ok(ish(num.mu, 5.455,0.001), "loadmu")
552     ok(ish(num.sd, 1.701,0.001), "loadsds") end
553
554 function go.egsShow( t)
555     oo(Egs{"name","Age","Weigh-"}) end
556
557 function go.egsHead( )
558     ok(Egs({"name","age","Weight!").cols.x, "Egs") end
559
560 function go.egs( eggs)
561     eggs = csv2egs(the.file)
562     ok(ish(egs.cols.x[1].mu, 5.455,0.001),"loadmu")
563     ok(ish(egs.cols.x[1].sd, 1.701,0.001),"loadsds") end
564
565 -----
566 the = settings(help)
567 go.main(the.todo, the.seed)
568 os.exit(go.fail)

```