

```

1  #!/usr/bin/env lua
2  -- vim: ts=2 sw=2 et:
3  -- (c) 2022, Tim Menzies
4  -- Usage of the works is permitted provided that this instrument is
5  -- retained with the works, so that any entity that uses the works is
6  -- notified of this instrument.  DISCLAIMER: THE WORKS ARE WITHOUT WARRANTY.
7
8  local b4={}; for k,v in pairs(_ENV) do b4[k]=k end
9  local help = {}
10
11  gate: explore the world better, explore the world for good.
12  (c) 2022, Tim Menzies
13
14
15  Ba 56 Bad <----- planning= (better - bad)
16  Be 4 Better monitor = (bad - better)
17
18
19
20
21
22  OPTIONS (inference control):
23  -far real limit for search for far points = .9
24  -l int dist. co-efficient. Euclidean if 2 = 2
25  -k int Bayes: handle rare classes = 2
26  -m int Bayes: handle rare values = 1
27  -min real min size = .5
28  -seed int random number seed = 10019
29  -keep int numbers to keep per column = 512
30  -wait int pre-learning, wait a few examples = 5
31
32  OTHER:
33  -h show help = false
34  -dump enable stack dump on failures = false
35  -file file with data = ../etc/data/auto93.csv
36  -rnd str pretty print control for floats = %5.3f
37  -todo str start-up action = the
38
39  EXAMPLES:
40  lua gate.lua -todo list : list all actions
41  lua gate.lua -todo all : run all actions
42  ]]
43
44  -----
45
46  -- define the local names
47  local the,go,no,fails = {}, {}, {}, 0
48  local abs,updates,cli,coerce,copy,svg,demos,ent,fu,fmt,fmt2,gt,inc,last,log
49  local lt,map,map2,max,merge,merges,min,new,o,ok,obj,oo,ooo,per,push
50  local r,rnd,rnds,sd,settings,slots,sort,splice,sum
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  -- maths
87  rs = math.random
88  abs = math.abs
89  log = math.log
90  min = math.min
91  max = math.max
92  function ent(t, n,e)
93  n=0; for _,v in pairs(t) do n=n+v end
94  e=0; for _,v in pairs(t) do e=e-v*n*log(v/n,2) end; return e end
95
96  function per(t,p) return t[ ((p or .5)*#t) // 1 ] end
97
98  function sd(sorted,f, ninety,ten)
99  if #sorted <= 10 then return 0 end
100  ninety,ten = per(sorted,.90), per(sorted,.10)
101  if f then ninety,ten = f(ninety), f(ten) end
102  return (ninety-ten)/2.564 end -- 2*(1.2 + 0.1*(0.9-0.88493)/(0.9032-0.88493))
103
104  -- lists
105  function last(t) return t[#t] end
106  function inc(f,a,n) f=f or(); f[a]=(f[a] or 0) + (n or 1) return f end
107  function push(t,x) t[1 + #t] = x; return x end
108  function sort(t,f) table.sort(t,f); return t end
109  function map(t,f, u) u={}; for _,v in pairs(t) do u[1+#u]=f(v) end;return u end
110  function map2(t,f, u) u={}; for k,v in pairs(t) do u[k] = f(k,v) end;return u end
111
112  function copy(t, u)
113  if type(t) ~= "table" then return t end
114  u={}; for k,v in pairs(t) do u[copy(k)]=copy(v) end
115  return setmetatable(u,getmetatable(t)) end
116
117  function slots(t, u,public)
118  function public(k) return tostring(k):sub(1,1) ~= " " end
119  u={}; for k,v in pairs(t) do if public(k) then u[1+#u]=k end end
120  return sort(u) end
121
122  function splice(t,lo,hi, j,u)
123  lo, hi = lo or 1, hi or #t
124  u={}; for j=lo,hi do u[1+#u]=t[j] end; return u end
125
126  -- things to strings
127  fmt= string.format
128  fmt2= function(k,v) return fmt("%.5s %s",k,v) end
129
130  function ooo(t) print( #t>1 and o(t) or oo(t)) end
131  function o(t,s) return "["..table.concat(map(t,tostring),s or ",",).."]" end
132  function oo(t,sep, slot)
133  function slot(k) return fmt2(k, t[k]) end
134  return (t.is or "")..o(map(slots(t),slot),sep or " ") end
135
136  function rnds(t,f) return map(t, function(x) return rnd(x,f) end) end
137  function rnd(x,f)
138  return fmt(type(x)=="number" and (x-x//1 and f or the.rnd) or "%s",x) end
139
140  -- strings to things
141  function coerce(x)
142  x = x:match("%%(%-%)%s*")
143  if x=="true" then return true elseif x=="false" then return false end
144  return math.tointeger(x) or tonumber(x) or x end
145
146  function csv(src, things)
147  function things(s, t)
148  t={}; for y in s:gmatch("[^,]+") do t[1+#t]=coerce(y) end; return t end
149  src = io.input(src)
150  return function(x) x=io.read()
151  if x then return things(x) else io.close(src) end end end
152
153  -- misc
154  function fu(x) return function(t) return t[x] end end
155  function lt(t) return function(t,u) return t[x] < u[x] end end
156  function gt(x) return function(t,u) return t[x] > u[x] end end
157
158  function updates(obj,data)
159  if type(data) ~= "string"
160  then for row in csv(data) do obj:update(row) end
161  else for _,x in pairs(data or {}) do obj:update(x) end end
162  return obj end
163
164  function merge(i,j, k)
165  k = i + j
166  if k:div()*.95 <= (i.n*i:div() + j.n*j:div())/k.n then return k end end
167
168  function merges(b4, a,b,c,j,n,tmp)
169  j,n,tmp = 1,#b4,{}
170  while j<#n do
171  a, b = b4[j], b4[j+1]
172  if b then
173  c = merge(a,b)
174  if c then a, j = c, j+1 end end
175  tmp[#tmp+1] = a
176  j = j+1 end
177  return #tmp==#b4 and tmp or merges(tmp) end
178
179  -- startup, execution, unit tests
180  function settings(t,help)
181  help:gsub("[^ ]-[^(%s+)](%s+)[^%n]*%s([%^s]+)",function(k,x) t[k]=coerce(x) end)
182  return t end
183
184  function cli(the, flag)
185  for k,v in pairs(the) do
186  flag="--."..k
187  for n,flag1 in ipairs(arg) do
188  if flag1 == flag then
189  v = v==false and "true" or v==true and "false" or arg[n+1]
190  the[k] = coerce(v) end end end
191  if the.h then os.exit(print(help)) else return the end end
192
193  function ok(test,msg)
194  print("", test and "PASS" or "FAIL", msg or "")
195  if not test then
196  fails= fails+1
197  if the.dump then assert(test,msg) end end end
198
199  function demos(the,go, demol,defaults)
200  function demol(txt,f)
201  assert(f:fmt("unknown start-up action: %s ",txt))
202  the = copy(defaults)
203  math.randomseed(the.seed or 10019)
204  print(txt)

```

```

205  f()
206  end -----
207  defaults = copy(the)
208  if the.todo=="all"
209  then for _,txt in pairs(slots(go)) do
210  demol(txt, go[txt]) end
211  else demol(the.todo, go[the.todo]) end end
212
213  -- classes
214  function new(klass,...)
215  local obj = setmetatable({},klass)
216  local res = klass.new(obj,...)
217  if res then obj = setmetatable(res,klass) end
218  return obj end
219
220  function obj(name, t)
221  t=({__tostring=oo, is=name or ""}); t.__index=t
222  return setmetatable(t, ({__call=new}) end

```

```

223 -----
224 local Some,Sym,Num,Bin = obj"Some", obj"Sym", obj"Num", obj"Bin"
225 local Clus,Egs,Nb,Abcd = obj"Cls", obj"Egs", obj"Nb", obj"Abcd"
226 local Cluster = obj"Cluster"
227 -----
228 function Bin:new(at,name, lo,hi,ys)
229   self.at, self.name = at or 0, name or ""
230   self.lo, self.hi, self.ys = lo, hi or lo, ys or Sym() end
231
232 function Bin:_tostring()
233   local x,lo,hi,big = self.name, self.lo, self.hi, math.huge
234   if lo == hi then return fmt("%s<=%s",x, lo)
235   elseif hi == big then return fmt("%s<=%s",x, lo)
236   elseif lo == -big then return fmt("%s<=%s",x, hi)
237   else return fmt("%s<=%s<%s",lo,x,hi) end end
238
239 function Bin:select(row)
240   local x, lo, hi = row[self.at], self.lo, self.hi
241   return x=="?" or lo == hi and lo == x or lo <= x and x < hi end
242
243 function Bin:update(x,y)
244   if x<self.lo then self.lo = x end
245   if x>self.hi then self.hi = x end
246   self.ys:update(y) end
247
248 function Bin:div() return self.ys:div() end
249
250 function Bin:_add(other)
251   return Bin(self.at, self.name, self.lo, after.hi, self.ys + other.ys) end
252 -----
253 function Sym:new(at,name)
254   self.at, self.name = at or 0, name or ""
255   self.n, self.has, self.mode, self.most = 0, {},nil,0 end
256
257 function Sym:update(x,inc)
258   if x ~= "?" then
259     inc = inc or 1
260     self.n = self.n + inc
261     self.has[x] = inc + (self.has[x] or 0)
262     if self.has[x] > self.most then self.most,self.mode = self.has[x],x end end
263   return x end
264
265 function Sym:mid() return self.mode end
266 function Sym:div() return ent(self.has) end
267
268 function Sym:like(x,prior)
269   return ((self.has[x] or 0) + the.m*prior)/(self.n + the.m) end
270
271 function Sym:dist(x,y)
272   return x=="?" and y=="?" and 1 or x==y and 0 or 1 end
273
274 function Sym:_add(other, out)
275   out=Sym(self.at,self.name)
276   for x,n in pairs(self.has) do out:update(x,n) end
277   for x,n in pairs(other.has) do out:update(x,n) end
278   return out end
279
280 function Sym:bins(other)
281   local out = {}
282   local function known(x) out[x] = out[x] or Bin(self.at, self.name, x,x) end
283   for x,n in pairs(self.has) do known(x); out[x].ys:update("left", n) end
284   for x,n in pairs(other.has) do known(x); out[x].ys:update("right", n) end
285   return map(slots(out), function(k) return out[k] end) end
286 -----
287 function Some:new()
288   self.kept, self.ok, self.n = {}, false,0 end
289
290 function Some:update(x, a)
291   self.n = 1 + self.n
292   a = self.kept
293   if #a < the.keep then self.ok=false; push(a,x)
294   elseif r() < the.keep/self.n then self.ok=false; a[r(#a)]=x end end
295
296 function Some:has()
297   if not self.ok then table.sort(self.kept) end
298   self.ok = true
299   return self.kept end
300 -----
301 function Num:new(at,name)
302   self.at, self.name = at or 0, name or ""
303   self.w = self.name:find"-$" and -1 or 1
304   self.some=Some()
305   self.n,self.mu,self.m2,self.sd,self.lo,self.hi = 0,0,0,0,1E32,-1E32 end
306
307 function Num:update(x,_, a,d)
308   if x ~= "?" then
309     self.some:update(x)
310     self.n = self.n + 1
311     self.lo = min(x, self.lo)
312     self.hi = max(x, self.hi)
313     d = x - self.mu
314     self.mu = self.mu + d/self.n
315     self.m2 = self.m2 + d*(x - self.mu)
316     self.sd = (self.m2<0 or self.n<2) and 0 or ((self.m2/(self.n - 1))^0.5) end
317   return x end
318
319 function Num:_add(other, out)
320   out=Num(self.at,self.name)
321   for _,x in pairs(self.some.kept) do out:update(x) end
322   for _,x in pairs(other.some.kept) do out:update(x) end
323   return out end
324
325 function Num:mid() return self.mu end
326 function Num:div() return self.sd end
327
328 function Num:like(x,_)
329   local z, e, pi = 1E-64, math.exp(1), math.pi
330   if x < self.mu - 4*self.sd then return 0 end
331   if x > self.mu + 4*self.sd then return 0 end
332   return e^(-(x - self.mu)^2 / (z + 2*self.sd^2)) / (z + (pi*2*self.sd^2)^.5) end
333
334 function Num:dist(x,y)
335   if x=="?" and y=="?" then return 1 end
336   if x=="?" then y = self:norm(y); x = y<.5 and 1 or 0
337   elseif y=="?" then x = self:norm(x); y = x<.5 and 1 or 0
338   else x,y = self:norm(x), self:norm(y) end
339   return abs(x - y) end
340
341 function Num:norm(x, lo,hi)
342   lo,hi = self.lo, self.hi

```

```

343   return x=="?" and x or hi-lo < 1E-9 and 0 or (x - lo)/(hi - lo) end
344
345 function Num:bins(other, tmp,out,now,epsilon,minSize)
346   tmp = {}
347   for _,x in pairs(self.some.kept) do push(tmp, {x=x, y="left"}) end
348   for _,x in pairs(other.some.kept) do push(tmp, {x=x, y="right"}) end
349   tmp = sort(tmp,lt"x") -- ascending on x
350   out = {}
351   now = push(out, Bin(self.at, self.name, tmp[1].x))
352   epsilon = sd(tmp,fu"x") * the.cohen
353   minSize = (#tmp)*the.leaves
354   for j,xy in pairs(tmp) do
355     if j > minSize and j + minSize < #tmp then -- leave enough for other bins
356       if now.ys.n > minSize then -- enough in this bins
357         if xy.x ~= tmp[j+1].x then -- there is a break in the data
358           if now.hi - now.lo > epsilon then -- "now" not trivially small
359             now = push(out, Bin(self.at, self.name, now.hi)) end end end end
360           now:update(xy.x, xy.y) end
361         out[j].lo = -math.huge
362         last(out).hi = math.huge
363       return merges(out) end

```

```

364 -----
365 function Cols:new(names, col)
366   self.names, self.all, self.x, self.y, self.klass = names, {}, {}, {}, nil
367   for at,name in pairs(names) do
368     col = push(self.all, (name:find"^[A-Z]" and Num or Sym)(at,name))
369     if not name:find"$" then
370       if name:find"$" then self.klass=col end
371       col.indep = not name:find"[+!$]"
372       push(col.indep and self.x or self.y, col) end end end
373

```

```

373 -----
374 function Egs:new() self.rows, self.cols = {},nil end
375
376 function Egs:clone(data)
377   return updates(Egs()::update(self.cols.names), data) end
378
379 function Egs:update(row, add)
380   add = function(col) col:update(row[col.at]) end
381   if self.cols
382     then map(self.cols.all,add); push(self.rows, row)
383   else self.cols = Cols(row) end
384   return self end
385
386 function Egs:mid(cols)
387   return map(cols or self.cols.y, function(col) return col:mid() end) end
388
389 function Egs:div(cols)
390   return map(cols or self.cols.y, function(col) return col:div() end) end
391
392 function Egs:like(row,egs,overall, prior,like,col)
393   prior = (#self.rows + the.k) / (overall + the.k * #egs)
394   like = log(prior)
395   for at,x in pairs(row) do
396     col = self.cols.all[at]
397     if x == "" and col.indep then like=like + log(col:like(x,prior)) end end
398   return like end
399
400 function Egs:klass(row) return row[self.cols.klass.at] end
401
402 function Egs:better(row1,row2)
403   local s1, s2, n, e = 0, 0, #self.cols.y, math.exp(1)
404   for _,col in pairs(self.cols.y) do
405     local a = col:norm(row1[col.at])
406     local b = col:norm(row2[col.at])
407     s1 = s1 - e*(col.w * (a - b) / n)
408     s2 = s2 - e*(col.w * (b - a) / n) end
409   return s1 / n < s2 / n end
410
411 function Egs:betters()
412   return sort(self.rows, function(a,b) return self:better(a,b) end) end
413
414 function Egs:dist(row1,row2, d,n)
415   d,n = 0, #self.cols.x
416   for _,col in pairs(self.cols.x) do
417     d = d + col:dist(row1[col.at], row2[col.at])^the.1 end
418   return (d/n)^(1/the.1) end
419
420 function Egs:around(row1, rows)
421   function around(row2) return (dist=self:dist(row1,row2),row=row2) end
422   return sort(map(rows or self.rows,around), lt="dist") end
423
424 function Egs:far(row, rows)
425   return per(self:around(row, rows or many(self.rows, the.some))).row end
426
427 function Egs:halves(top, here)
428   top = top or self
429   here = Halved(eg,top)
430   if here.lefts and here.lefts.rows < #eg.rows then
431     here.lefts = here.lefts:halves(top)
432     here.rights = here.rights:halves(top) end
433   return here end
434
435 function Egs:bestsRests( rests, keep, run, b4)
436   function run(eg,b4, here)
437     here = Halved(eg, top, b4)
438     if here.lefts and here.lefts.rows < #eg.rows
439       then map(here.rights.rows,
440         function(r) if r()<keep then rests:update(r) end end)
441     else return eg, rests end
442   end -----
443   rests = self:clone()
444   keep = (#self.rows)^the.min
445   keep = the.keep*keep / (#self.rows - keep)
446   b4 = self:far(any(self.rows))
447   return run(self, b4) end
448
449

```

```

449
450 function Halved:new(eg,top,b4, rows,some)
451   self.top = top or eg
452   self.eg = eg
453   rows = self.eg.rows
454   if #eg.rows >= (#top.rows)^the.min then
455     some = many(rows, the.some)
456     self.left = b4 or top:far(any(some), some)
457     self.right = top:far(self.left, some)
458     self.c = self.top:dist(self.left, self.right)
459     if b4 and eg:better(right,left) then self.left end
460     self.left, self.right = self.right, self.left end
461     self.lefts = self.eg:clone()
462     self.rights = self.eg:clone()
463     for n,projection in pairs(self:projections(rows)) do
464       C,b,a = #rows/2 and self:lefts or self:rights:update(projection.row) end
465       self.gaurd = self.top:dist(left, last(left.rows)) end
466     return self end
467
468 function Halved:projections(rows)
469   return sort(map(rows, function(r) return self:project(r) end), lt="x") end
470
471 function Halved:project(row, z,a,b,c)
472   z = 1/math.huge
473   C,b,a = self.c, self.top:dist(row,self.right), self,top:dist(row,self.left)
474   return (x = (a^2 + c^2 - b^2) / (2*c + z),
475     row = row) end
476 -----
477 function Nb:new()
478   self.all, self.some, self.log = nil, {}, {} end
479
480 function Nb:update(row)
481   if self.all
482     then if #self.all.rows > the.wait then
483         push(self.log, ( want = self.all:klass(row),
484           got = self:classify(row) )) end
485     else self.all = Egs():update(row) end end
486
487 function Nb:train(row, k)
488   k = self.all:klass(row)
489   self.some[k] = self.some[k] or self.all:clone()
490   self.some[k]:update(row)
491   self.all:update(row) end
492
493 function Nb:classify(row, most,klass,tmp,out)
494   most = -math.huge
495   for klass,eg in pairs(self.some) do
496     out = out or klass
497     tmp = eg:like(row, self.some, #self.all.rows)
498     if tmp > most then most,out = tmp, klass end end
499   return out,most end
500 -----
501 function Egs:tree(other,min, kids,score)
502   function gain(col1, col2, all, sum,bins)
503     sum = 0
504     bins = coll:bins(col2)
505     map(bins, function(bin)
506       bin.here = self
507       bin.has = {self:clone(),self:clone()}
508       sum = sum + bin.ys:n/all * bin.ys:div() end)
509     return (bins=bins, gain=sum)
510   end -----
511   n = #self.rows + #other.rows
512   stop = stop or n^the.min
513   if n < stop
514     then return self
515   else cols = map2(self.col.x, function(at,col)
516     return (w=gain(col, other.col.x[at], n), col=col) end)
517     bins = sort(cols,fu="w")[1].bins
518     for at,eg in pairs(self.w[other]) do
519       for _,row in pairs(eg.rows) do
520         for _,bin in pairs(bins) do
521           sub = bin.has[at]
522           if bin:select(row) then sub:update(row); break end end end end
523         self.kids = map(bins,
524           function(bin) bin.kid = bin.has[1]:tree(bin.has[2]) end) end end
525     -- XXXX not done yet. need to return the ocal kids

```

```

527 -----
528 function Abcd:new(data,rx)
529   self.data, self.rx = data or "", rx or ""
530   self.yes, self.no = 0,0
531   self.known, self.a, self.b, self.c, self.d = {},(),(),(),{} end
532
533 function Abcd:exists(x, new)
534   new = not self.known[x]
535   inc(self.known,x)
536   if new then
537     self.a[x]=self.yes + self.no; self.b[x]=0; self.c[x]=0; self.d[x]=0 end end
538
539 function Abcd:report( p,out,a,b,c,d,dp,pf,pn,f,acc,g,prec)
540   p = function (z) return math.floor(100*z + 0.5) end
541   out= {}
542   for x,xx in pairs( self.known ) do
543     dp,pf,pn,prec,g,f,acc = 0,0,0,0,0,0,0
544     a= (self.a[x] or 0); b= (self.b[x] or 0);
545     c= (self.c[x] or 0); d= (self.d[x] or 0);
546     if b+d > 0 then pd = d / (b+d)
547     if a+c > 0 then pf = c / (a+c) end
548     if a+c > 0 then pn = (b+d) / (a+c) end
549     if c+d > 0 then prec = d / (c+d) end
550     if 1-pf+pd > 0 then g=2*(1-pf) * pd / (1-pf+pd) end
551     if prec+pd > 0 then f=2*prec*pd / (prec + pd) end
552     if self.yes + self.no > 0 then
553       acc= self.yes / (self.yes + self.no) end
554     out[x] = {data=self.data,rx=self.rx,num=self.yes+self.no,
555       a=a,b=b,c=c,d=d,acc=p(acc),
556       prec=p(prec), pd=p(pd), pf=p(pf), f=p(f), g=p(g), class=x) end
557   return out end
558
559 function Abcd:pretty(t, s1,s2,d,s,u)
560   print""
561   s1 = "%10s| %10s| %4s| %4s| %4s| %4s "
562   s2 = "%13s| %3s| %3s| %4s| %3s| %3s|"
563   d,s = "-----", (s1 .. s2)
564   print (fmt (s,"db","rx","a","b","c","d","acc","pd","pf","prec","f","g"))
565   print (fmt (s,d,d,d,d,d,d,d,d,d,d,d,d,d,d))
566   for x,u in pairs(sort(map(t,function(x) return x end),
567     function(a,b) return (a.b+a.d>b.b+b.d) end)) do
568     print (fmt (s.." %s", u.data,u.rx,u.a, u.b, u.c, u.d,
569       u.acc, u.pd, u.pf, u.prec, u.f, u.g, u.class)) end end
570
571 function Abcd:adds(gotwants, show)
572   for key,one in pairs(gotwants) do
573     self:exists(one.want)
574     self:exists(one.got)
575     if one.want == one.got then self.yes=self.yes+1 else self.no=self.no+1 end
576     for x,xx in pairs(self.known) do
577       if one.want == x
578         then inc(one.want == one.got and self.d or self.b, x)
579       else inc(one.got == x
580         and self.c or self.a, x) end end end
581   return show and self:pretty(self:report()) or self:report() end

```

```

581 -----
582 function go.list()
583   map(slots(go), function(x) print(fmt("lua gate.lua -todo %s", x)) end) end
584
585 function go.the() ooo(the) end
586
587 function go.sort( t)
588   t={10,9,3}
589   ooo(sort(t)) end
590
591 function go.ent() ok(abs(1.3788 - ent(a=4,b=2,c=1)) < 0.001,"enting") end
592
593 function go.ooo() ooo{cc=1,bb={ff=4,dd=5,bb=6}, aa=3} end
594
595 function go.copy( t,u)
596   t = {a=1,b=2,c={d=3,e=4,f={g=5,h=6}}}
597   u = copy(t)
598   t.c.f.g = 100
599   ok(u.c.f.g ~= t.c.f.g, "deep copy") end
600
601 function go.rnds() ooo(rnds(3.421212, 10.1121, 9.1111, 3.44444)) end
602
603 function go.csv( n)
604   n=0; for row in csv(the.file) do n=n+1 end; ok(n==399,"stuff") end
605
606 function go.some( s)
607   the.keep = 64
608   s = Some(); for i=1,10^6 do s:update(i) end
609   ooo(s:has()) end
610
611 function go.num( n,mu,sd)
612   n, mu, sd = Num(), 10, 1
613   for i=1,10^3 do
614     n:update(mu + sd*math.sqrt(-2*math.log(r()))*math.cos(2*math.pi*r())) end
615   ok(abs(n:mid() - mu) < 0.025, "sd")
616   ok(abs(n:div() - sd) < 0.05, "div") end
617
618 function go.updates( n)
619   print(updates(Num(), {1,2,3,4,5}) + updates(Num(), {11,12,13,14,15}))
620   end
621
622 function go.sym( s,mu,sd)
623   s= Sym()
624   for i=1,100 do
625     for k,n in pairs(a=4,b=2,c=1) do s:update(k,n) end end
626   ooo(s:has) end
627
628 function go.egs(f)
629   for _,col in pairs(updates(Egs(),f or "../etc/data/diabetes.csv").cols.all) do
630     print("%u",col) end end
631
632 function go.clone(f, a,b)
633   a = updates(Egs(),f or "../etc/data/diabetes.csv")
634   b = a:clone(a.rows)
635   print(a.cols.x[1].sd)
636   print(b.cols.x[1].sd)
637   ok(a.cols.x[1].sd == b.cols.x[1].sd, "same y") end
638
639 function go.abcd()
640   local t={}
641   for _ = 1,6 do push(t,{want="yes",got="yes"}) end
642   for _ = 1,2 do push(t,{want="no",got="no"}) end
643   for _ = 1,6 do push(t,{want="maybe",got="maybe"}) end
644   for _ = 1,1 do push(t,{want="maybe", got="no"}) end
645   Abcd():adds(t,true) end
646
647 function go.nb(f, nb)
648   nb = updates(Nb(),f or "../etc/data/diabetes.csv")
649   Abcd():adds(nb.log, true) end
650
651 function go.nhsb()
652   go.nb("../etc/data/soybean.csv") end
653
654 function go.bestrest( eg,best,rest,rows,n)
655   eg= updates(Egs(), "../etc/data/auto93.csv")
656   rows = eg:betters()
657   n = (#rows)^.5 // 1
658   best = splice(rows, 1,n)
659   rest = eplice(rows, #rows-n)
660   best = eg:clone(best)
661   rest = eg:clone(rest)
662   ooo(rnds(best:mid()))
663   ooo(rnds(rest:mid()))
664   end
665
666 -----
667 the = settings(the,help)
668
669 if pcall(debug.getlocal, 4, 1)
670 then return {Num=Num, Sym=Sym, Egs=Egs} -- called as sub-module. return classes
671 else the = cli(the) -- update 'the' from command line
672   demos(the,go) -- run some demos
673   for k,v in pairs(_ENV) do if not b4[k] then print("?",k,type(v)) end end
674   os.exit(fails) end

```