

```

1  #!/usr/bin/env lua
2
3  --
4  --
5  -- a little lite
6  -- ZWA learning
7  -- library
8  --
9  --
10 --
11 --
12 --
13 --
14 --
15 --
16 --
17 --
18 local it=require"options"
19 what = "Small sample multi-objective optimizer.",
20 who = "(c) 2021 Tim Menzies <timm@ieee.org> unlicense.org",
21 why = {}
22 Sort N examples on multi-goals using a handful of 'hints'; i.e.
23
24 - Evaluate and rank, a few examples (on their y-values);
25 - Sort other examples by x-distance to the ranked ones;
26 - Recurse on the better half (so we sample more and more
27   from the better half, then quarter, then eighth...).
28
29 A regression tree learner then explores the examples (sorted
30 left to right, worst to best). By finding branches that
31 reduce the variance of the index of those examples, this
32 tree reports what attribute ranges select for the better (or
33 worse) examples.  ]],
34
35 how={{"FILE",      "-f",      ".data/auto93.csv",    "read data from file"},
36       {"CULL",     "-c",      .5,      "cuts per generation"},
37       {"HELP",     "-h",      false,    "show help"},
38       {"HINTS",    "-H",      4,      "hints per generation"},
39       {"P",        "-p",      2,      "distance calc exponent"},
40       {"TINY",     "-t",      .5,      "div list into 'small'"},
41       {"SEED",     "-S",      10019,   "random number seed"},
42       {"TRAIN",    "-t",      .5,      "size of training set"},
43       {"TODO",     "-T",      "all",    "run unit test, or 'all'"},
44       {"TRIVIAL",  "-v",      .35,    "small delta=trivial*sd"},
45       {"WILD",     "-W",      false,   "run tests, no protection"}},
46
47 local __require__lib"
48 local abs,bchop,cut,copy = __abs, __bchop, __cat, __copy
49 local csv,first,firsts,fmt,has = __csv, __first, __firsts, __fmt, __has
50 local keys,last,lap,map,obj = __keys, __last, __lap, __map, __obj
51 local out,pop,push,rand,shout = __out, __pop, __push, __rand, __shout
52 local rnd,rnds,roques,second = __rnd, __rnds, __roques, __second
53 local shuffle,sort,sum,top = __shuffle, __sort, __sum, __top
54
55 --[[
56 Spans
57   Little languages:
58   - options
59   - data language
60
61 Lesson plan
62 -- w1: ssystems: github. github workplaces. unit tests. doco tools.
63 -- w2: num, sym
64 -- w3: sample
65 -- w4: eval, knn, unfairness
66 -- w5:
67 --]]

```

```

68 -- NUM -----
69 --
70 --
71 -- ## Stuff for tracking 'Num'bers.
72 -- 'Num's track a list of number, and can report it sorted.
73 local Num=obj"Num"
74 function Num.new(inits,at,txt, self)
75   self=has(Num,{at=at or 0, txt=txt or "", w=(txt or ""):find"--" and -1 or 1,
76     has={}, n=0, lo=1E32, hi =1E-32, ready=true})
77   for _,one in pairs(inits or {}) do self:add(one) end
78   return self end
79
80 function Num:add(x)
81   if x>self.hi then self.hi = x
82   elseif x<self.lo then self.lo = x end
83   push(self.has,x); self.n=self.n+1; self.ready=false end
84
85 -- Ensure that the returned list of numbers is sorted.
86 function Num:all(x)
87   if not self.ready then table.sort(self.has) end
88   self.ready = true
89   return self.has end
90
91 function Num:dist(a,b)
92   if a=="?" then b=self:norm(b); a = b>.5 and 0 or 1
93   elseif b=="?" then a=self:norm(a); b = a>.5 and 0 or 1
94   else a,b = self:norm(a), self:norm(b) end
95   return abs(a-b) end
96
97 -- Combine two 'num's.
98 function Num:merge(other, new)
99   new = Num()
100   new.at, new.txt = self.at, self.txt
101   for _,x in pairs(self.has) do new:add(x) end
102   for _,x in pairs(other.has) do new:add(x) end
103   return new end
104
105 -- The 'mid' is the 50th percentile.
106 function Num:mid() return self:per(.5) end
107
108 -- Return 'x' normalized 0..1, lo..hi.
109 function Num:norm(x, lo,hi)
110   if x=="?" then return x end
111   lo,hi = self.lo, self.hi
112   return abs(hi - lo) < 1E-32 and 0 or (x - lo)/(hi - lo) end
113
114 -- Return the 'p'-th percentile number.
115 function Num:per(p, t)
116   t = self:all()
117   p = p*#t//1
118   return #t<2 and t[1] or t[p < 1 and 1 or p>#t and #t or p] end
119
120 -- The 10th to 90th percentile range is 2.56 times the standard deviation.
121 function Num:sd() return self:per(.9) - self:per(.1))/ 2.56 end
122 function Num:spread() return self:sd() end
123
124 -- Create one span (each has the row indexes of the rows)
125 -- where each span has at least 'tiny' items and span is more than
126 -- 'trivially' small.
127 local div -- defined below
128 function Num:spans(sample,tiny,trivial)
129   local xys = {}
130   for _,eg in pairs(sample.egs) do
131     local x = eg[self.at]
132     if x == "?" then push(xys, {col=col, x=x, y=eg[sample.klass.at]}) end end
133   return div(xys, tiny, trivial, self, getmetatable(sample.klass)) end
134
135 -- SYM -----
136 --
137 --
138 -- Stuff for tracking 'Sym'bol Counts.
139 -- 'Sym's track symbol counts and the 'mode' (most frequent symbol).
140 local Sym=obj"Sym"
141 function Sym.new(inits,at,txt, self)
142   self=has(Sym,{at=at or 0, txt=txt or "", has={}, n=0, mode=nil, most=0})
143   for _,one in pairs(inits or {}) do self:add(one) end
144   return self end
145
146 function Sym:add(x,n)
147   n = n or 1
148   self.n = self.n + n
149   self.has[x] = n + (self.has[x] or 0)
150   if self.has[x] > self.most then self.most, self.mode = self.has[x], x end end
151
152 function Sym:dist(a,b) return a==b and 0 or 1 end
153
154 function Sym:merge(other)
155   new=Sym()
156   new.at, new.txt = self.at, self.txt
157   for k,n in pairs(self.has) do new:add(k,n) end
158   for k,n in pairs(other.has) do new:add(k,n) end
159   return new end
160
161 function Sym:mid() return self.mode end
162
163 -- Create one span holding row indexes associated with each symbol
164 function Sym:spans(sample,...)
165   local xys,yklass = {}, getmetatable(sample.klass)
166   for pos,eg in pairs(sample.egs) do
167     local x = eg[self.at]
168     if x == "?" then
169       xys[x] = xys[x] or yklass()
170       xys[x]:add(eg[sample.klass.at]) end end
171   return map(xys, function(x,ys) return {col=self, lo=x, hi=x, has=ys} end) end
172
173 function Sym:spread()
174   return sum(self.has,
175     function(nl) return -nl/self.n * math.log(nl/self.n,2) end) end
176
177 -- SKIP -----
178 --
179 -- ## Stuff for skipping all things sent to a column
180 local Skip=obj"Skip"
181 function Skip.new(_,at,txt) return has(Skip,{at=at or 0, txt=txt or "", n=0}) end
182 function Skip:add(x) self.n = self.n + 1; return x end
183 function Skip:mid() return "?" end

```

Dec 13, 21 9:32

l5.lua

Page 3/7

```

184 -- SAMPLE -----
185
186 -- Samples store examples. Samples know about
187 -- (a) lo,hi ranges on the numerics
188 -- and (b) what are independent 'x' or dependent 'y' columns.
189 local Sample = {}
190 function Sample.new(      src,self)
191   self = has(Sample,{names=nil, klass=nil, all={}, ys={}, xs={}, eggs={})
192   if src then
193     if type(src)=="string" then for x in csv(src) do self:add(x) end end
194     if type(src)=="table" then for _,x in pairs(src) do self:add(x) end end end
195   return self end
196
197 function Sample:add(eg,      ako,what,xy)
198   if not self.names
199   then -- create the column headers
200     self.names = eg
201     for at,x in pairs(eg) do
202       ako = (x:find"." and Skip) or
203             (x:match"%[A-Z]" and Num) or
204             Sym
205       what = push(self.all, ako({}, at, x))
206       if not x:find"." then
207         if x:find"!" then self.klass = what end
208         xy = (x:find"%" or x:find"=") or x:find"!%" and self.ys or self.xs
209         push(xy, what) end end end
210   else -- store another example; update column headers
211     push(self.egs, eg)
212     for at,x in pairs(eg) do if x ~= "?" then self.all[at]:add(x) end end end
213   return self end
214
215 function Sample:better(eg1,eg2,      e,n,a,b,s1,s2)
216   n,s1,s2,e = #self.ys, 0, 0, 2.71828
217   for _,num in pairs(self.ys) do
218     a = num:norm(eg1[num.at])
219     b = num:norm(eg2[num.at])
220     s1 = s1 - e^(num.w * (a-b)/n)
221     s2 = s2 - e^(num.w * (b-a)/n) end
222   return s1/n < s2/n end
223
224 function Sample:betters(egs)
225   return sort(egs or self.egs,function(a,b) return self:better(a,b) end) end
226
227 function Sample:clone(      inits,out)
228   out = Sample.new({}):add(self.names)
229   for _,eg in pairs(inits or {}) do out:add(eg) end
230   return out end
231
232 function Sample:dist(eg1,eg2,      a,b,d,n,inc)
233   d,n = 0,0
234   for _,col in pairs(self.xs) do
235     a,b = eg1[col.at], eg2[col.at]
236     inc = a=="?" and b=="?" and 1 or col:dist(a,b)
237     d = d + inc*it.P
238     n = n + 1 end
239   return (d/n)^(1/it.P) end
240
241 -- Report mid of the columns
242 function Sample:mid(cols)
243   return lap(cols or self.ys,function(col) return col:mid() end) end
244
245 -- Return spans of the column that most reduces variance
246 function Sample:bestSplits(tiny, trivials)
247   local function xpect1(col,      total,xpect,spans,total,xpect)
248     local function xpect1(span) return span.has.n/total * span.has:spread() end
249     spans = col:spans(self, tiny,trivials[col.at])
250     total = sum(spans,function(span) return span.has.n end)
251     xpect = sum(spans,xpect1)
252     return {xpect, spans}
253   end
254   return first(sort(lap(self.xs, column1), firsts))[2] end
255
256 -- Split on column with best span, recurse on each split.
257 function Sample:tree(tiny,trivials,pre,      node,new,x)
258   pre=pre or ""
259   print(pre ..".".#self.egs)
260   tiny = tiny or (#self.egs)^it.TINY
261   trivials = trivials or map(self.xs,
262     function(_,x)
263       return x.at,it.TRIVIAL*x:spread() end)
264   node = {node=self, kids={}}
265   shout(egs=#self.egs, tiny=2*tinyx,)
266   if #self.egs <= 2*tiny then print(3333333);return node end
267   print("==")
268   for _,span in pairs(self:bestSplits(tiny,trivials)) do
269     new = self:clone()
270     for _,eg in pairs(self.egs) do
271       x = eg[span.col.at]
272       --print(span.col.at, x, span.lo, span.hi)
273       if x=="?" or (span.lo <= x and x <= span.hi) then new:add(eg) end end
274     --os.exit()
275     print(#new.egs)
276     push(node.kids, {txt = span.col.txt, txt= span.col.at,
277       lo = span.lo, hi = span.hi,
278       sub = new:tree(tiny,trivials,pre.."[".."]")}) end
279     --os.exit()
280   --end end
281   return node end
282
283 -- Find which leaf best matches an example 'eg':.w
284
285 function Sample:where(tree,eg,      max,x,default)
286   if #kid.has==0 then return tree end
287   max = 0
288   for _,kid in pairs(tree.node) do
289     if #kid.has > max then default,max = kid,#kid.has end
290     x = eg[kid.at]
291     if x == "?" then
292       if x <= kid.hi and x >= kid.lo then
293         return self:where(kid.has.eg) end end end
294   return self:where(default, eg) end
295
296 -- Discrimination -----
297
298 -- Input a list of {(x,y,...) values. Return spans that divide the 'x' values
299 -- to minimize variance on the 'y' values.
300 function div(xys, tiny, trivial,col,yklass)
301   xys = sort(xys, function(a,b) return a.x < b.x end)
302   local tenth=#xys/10
303   trival = trivial or it.TRIVIAL*(xys[9*tenth][1] - xys[tenth][1])/2.56
304   tiny = tiny or it.TINY*#xys
305   yklass = yklass or Num
306   local function mergeable(a,b)
307     new = a:merge(b)
308     b4 = (a.n*a:spread() + b.n*b:sd()) / new.n
309     if new:spread() <= b4 then return new end
310   end
311   local function merge(b4) -- merge adjacent spans if combo simpler to he parts
312     local j, tmp = 0, {}
313     while j < #b4 do
314       j = j + 1
315       local now, after = b4[j], b4[j+1]
316       if after then
317         local simpler = mergeable(now.has, after.has)
318         if simpler then
319           now = {col=col, lo=now.lo, hi= after.hi, has=simpler}
320           j = j + 1 end end
321       push(tmp,now) end
322     return #tmp==#b4 and b4 or merge(tmp) -- recurse until nothing merged
323   end
324   local function coverGaps(spans,      b4) -- cover gaps in number line
325     b4 = first(spans).hi
326     for _,span in pairs(spans) do span.lo=b4; b4=span.hi end
327     first(spans).lo = -math.huge
328     last(spans).hi = math.huge
329     return spans
330   end
331   local spans,span
332   span = {col=col,lo=xys[1].x, hi=xys[1].x, has=yklass()}
333   spans = {span}

```

Dec 13, 21 9:32

l5.lua

Page 4/7

```

334 for j,xy in pairs(xys) do
335   local x, y = xy.x, xy.y
336   if j < #xys - tiny and -- enough items remaining after split
337     x ~= xys[j+1].x and -- next item is different (so can split here)
338     span.has.n > tiny and -- span has enough items
339     span.hi - span.lo > trivial -- span is not trivially small
340   then span = push(spans, {col=col, lo=x, hi=x, has=yklass()}) -- then new span
341   end
342   span.hi = x
343   span.has:add(y) end
344 return merge(spans) end
345 --return coverGaps(merge(spans)) end
346

```

Dec 13, 21 9:32

l5.lua

Page 5/7

```

347 -- HINTING
348 --
349 -- Sorting on a few y values
350 local hints={}
351 function hints.default(eg) return eg end
352
353 function hints.sort(sample,scorefun, test,train,egs,scored,small)
354   sample = Sample.new(it.FILE)
355   train,test = {}, {}
356   for i,eg in pairs(shuffle(sample.egs)) do
357     push(i<= it.TRAIN*#sample.egs and train or test, eg) end
358   egs = copy(train)
359   small = (#egs)^it.TINY
360   local i=0
361   scored = {}
362   while #egs >= small do
363     local tmp={}
364     i = i + 1
365     io.stderr:write(fmt("%s",string.char(96+i)))
366     for j=1,it.HINTS do
367       egs[j] = (scorefun or hints.default)(egs[j])
368       push(tmp, push(scored, egs[j]))
369     end
370     egs = hints.ranked(scored,egs,sample)
371     for i=1,it.CULL*#egs//1 do pop(egs) end
372   end
373   io.stderr:write("\n")
374   train=hints.ranked(scored, train, sample)
375   return #scored, sample:clone(train), sample:clone(test) end
376
377 function hints.ranked(scored,egs,sample,worker, some)
378   function worker(eg) return {hints.rankOfClosest(scored,eg,sample),eg} end
379   scored = sample:betters(scored)
380   return lap(sort(lap(egs, worker),firsts),second) end
381
382 function hints.rankOfClosest(scored,egl,sample, worker,closest)
383   function worker(rank,eg2) return {sample:dist(egl,eg2),rank} end
384   closest = first(sort(map(scored, worker),firsts))
385   return closest[2] end --+ closest[1]/10^8 end
386

```

Dec 13, 21 9:32

l5.lua

Page 6/7

```

387 -- demos
388 --
389 it._eg={}
390 it._no={}
391 function it._eg.shuffle( t,u,v)
392   t={}
393   for i=1,32 do push(t,i) end
394   u = shuffle(copy(t))
395   v = shuffle(copy(t))
396   assert(#t == #u and u[1] ~= v[1]) end
397
398 function it._eg.lap()
399   assert(3==lap({1,2},function(x) return x+1 end)[2]) end
400
401 function it._eg.map()
402   assert(3==map({1,2},function(_,x) return x+1 end)[2]) end
403
404 function it._eg.tables()
405   assert(20==sort(shuffle({{10,20},{30,40},{40,50}}),firsts)[1][2]) end
406
407 function it._eg.csv( n,z)
408   n=0
409   for eg in csv(it.FILE) do n=n+1; z=eg end
410   assert(n==399 and z[#z]==50) end
411
412 function it._eg.rnds( t)
413   assert(10.2 == first(rnds({10.22,81.22,22.33},1))) end
414
415 function it._eg.sym( s)
416   s=Sym{"a","a","a","a","b","b","b","c"}
417   assert("a"==s.mode) end
418
419 function it._eg.num1( n)
420   n=Num(10,20,30,40,50,10,20,30,40,50,10,20,30,40,50)
421   assert(1.375 == n:norm(25))
422   assert(15.625 == n:sd()) end
423
424 function it._eg.sample( s,tmp,d1,d2,n)
425   s=Sample(it.FILE)
426   assert(2110 == last(s.egs)[s.all[4].at])
427   local sort1= s:betters(s.egs)
428   local lo, hi = s:clone(), s:clone()
429   for i=1,20 do lo:add(sort1[i]) end
430   for i=#sort1,#sort1-20,-1 do hi:add(sort1[i]) end
431   shout(s:mid())
432   shout(lo:mid())
433   shout(hi:mid())
434   for m,eg in pairs(sort1) do
435     n = bchop(sort1, eg,function(a,b) return s:better(a,b) end)
436     assert(m-n <=2) end end
437
438 function it._eg.dists( s,tmp,d1,d2,n)
439   s=Sample(it.FILE)
440   tmp = sort(lap(shuffle(s.egs),
441     function(eg2) return (s:dist(eg2,s.egs[1]), eg2) end),
442     firsts)
443   d1=s:dist(tmp[1][2], tmp[10][2])
444   d2=s:dist(tmp[1][2], tmp[#tmp][2])
445   assert(d1*10 < d2) end
446
447 function it._eg.hints( s,_,_,evals,sort1,train,test,n)
448   s = Sample(it.FILE)
449   evals, train,test = hints.sort(s)
450   test.egs = test:betters()
451   for m,eg in pairs(test.egs) do
452     n = bchop(train.egs, eg,function(a,b) return s:better(a,b) end); end end
453
454 function it._eg.dump()
455   shout(it) end
456 function it._eg.tree( s,t,u,egl,evals,ordered,rest)
457   s = Sample(it.FILE)
458   t = copy(s.names)
459   push(t,"Rank")
460   u = Sample.new():add(t)
461   evals, ordered,rest = hints.sort(s)
462   for m,eg in pairs(ordered.egs) do
463     egl = copy(eg)
464     push(egl,m)
465     u:add(egl) end
466   print(1)
467   u:tree() end
468
469 -- START-UP
470 --
471 it{demos=it._eg, nervous=true}
472

```

```
473 --[[
474   |  |  |
475   |  |  |
476
477 Spans
478 Little languages:
479   - options
480   - data language
481
482 Lesson plan
483 - w1: ssystems: github. github workplaces. unit tests. doco tools.
484
485 - w2: num, sym
486 - W3: sample
487 - w4: eval, knn, unfairnessness
488 - W5:
489
490 - seems to be a revers that i need to do .... but dont
491 - check if shuffle is working
492
493 teaching:
494 - sample is v.useful
495 --]]
```

```

1  local lib={}
2
3  --- ROGUES ---
4  ---
5  --- Call 'rogues', last thing, to find escaped locals.
6  lib._b4={}; for k,v in pairs(_ENV) do lib._b4[k]=k end
7  function lib.rogues()
8      for k,v in pairs(_ENV) do
9          if not lib._b4[k] then print("?rogue: ",k,type(v)) end end end
10
11  --- OBJECTS ---
12  ---
13  --- Create an instance
14  function lib.has(mt,x) return setmetatable(x,mt) end
15  --- Create a class
16  function lib.obj(s, o,new)
17      o = {__is=s, __tostring=lib.out}
18      o.__index = o
19      return setmetatable(o,{__call = function(_,...) return o.new(...) end}) end
20
21  --- RANDOM ---
22  ---
23  lib.Seed = 10019
24  --- random integers
25  function lib.randi(lo,hi) return math.floor(0.5 + lib.rand(lo,hi)) end
26  --- random floats
27  function lib.rand(lo,hi, mult,mod)
28      lo, hi = lo or 0, hi or 1
29      lib.Seed = (16807 * lib.Seed) % 2147483647
30      return lo + (hi-lo) * lib.Seed / 2147483647 end
31
32  --- MATHS ---
33  ---
34  lib.abs = math.abs
35  --- Round 'x' to 'd' decimal places.
36  function lib.rnd(x,d, n) n=10^(d or 0); return math.floor(x*n+0.5) / n end
37  --- Round list of items to 'd' decimal places.
38  function lib.rnds(t,d)
39      return lib.lap(t, function(x) return lib.rnd(x,d or 2) end) end
40
41  --- Sum items, filtered through 'f'.
42  function lib.sum(t,f, out)
43      f = f or function(x) return x end
44      out=0; for _,x in pairs(t) do out = out + f(x) end; return out end
45
46  --- FILES ---
47  ---
48  --- Return one table per line, split on commas.
49  function lib.csv(file, line)
50      file = io.input(file)
51      line = io.read()
52      return function()
53          if line then
54              t={}
55              for cell in line:gsub("[\r\n]", ""):gsub("#", ""):gmatch("[^\r\n,]+") do
56                  lib.push(t, tonumber(cell) or cell) end
57              line = io.read()
58              if #t>0 then return t end
59              else io.close(file) end end end
60
61  --- PRINTING ---
62  ---
63  lib.fmt = string.format
64  lib.say = function(...) print(lib.fmt(...)) end
65
66  --- Print as red, green, yellow, blue.
67  function lib.color(s,n) return lib.fmt("%27[1m27[%sm%s27]0m",n,s) end
68  function lib.red(s) return lib.color(s,31) end
69  function lib.green(s) return lib.color(s,32) end
70  function lib.yellow(s) return lib.color(s,34) end
71  function lib.blue(s) return lib.color(s,36) end
72
73  --- Printed string from a nested structure.
74  lib.shout = function(x) print(lib.out(x)) end
75  --- Generate string from a nested structures
76  --- (and don't print any contents more than once).
77  function lib.out(t,seen, u,key,value,public)
78      function key(k) return lib.fmt("%s", lib.blue(k), lib.out(t[k],seen)) end
79      function value(v) return lib.out(v,seen) end
80      if type(t) == "function" then return "(...)" end
81      if type(t) == "table" then return tostring(t) end
82      seen = seen or {}
83      if seen[t] then return "..." else seen[t] = t end
84      u = #t>0 and lib.lap(t, value) or lib.lap(lib.keys(t), key)
85      return lib.red((t._is or "").."["..lib.cat(u,"")..lib.red("]") end
86
87  --- TABLE ---
88  ---
89  --- Table to string.
90  lib.cat = table.concat
91  --- Return a sorted table.
92  lib.sort = function(t,f) table.sort(t,f); return t end
93  --- Return first, second, last item
94  lib.first = function(t) return t[1] end
95  lib.second = function(t) return t[2] end
96  lib.last = function(t) return t[#t] end
97  --- Function for sorting pairs of items.
98  lib.firsts = function(a,b) return a[1] < b[1] end
99  --- Add to end, pull from end.
100 lib.pop = table.remove
101 lib.push = function(t,x) table.insert(t,x); return x end
102
103 --- Random order of items in a list (sort in place).
104 function lib.shuffle(t, j)
105     for i=#t,2,-1 do j=lib.randi(1,i); t[i],t[j]=t[j],t[i] end; return t end
106
107 --- Collect values, passed through 'f'.
108 function lib.lap(t,f) return lib.map(t,f,1) end
109 --- Collect key, values, passed through 'f'.
110 --- If 'f' returns two values, store as key,value.
111 --- If 'f' returns one values, store at index value.
112 --- If 'f' return nil then add nothing (so 'map' is also 'select').
113 function lib.map(t,f,one)
114     u={}; for k,v in pairs(t) do
115         if one then x,y=f(y) else x,y=f(x,y) end
116         if x ~= nil then
117             if y then u[x]=y else u[1+#u]=x end end end
118     return u end
119
120 --- Shallow copy
121 function lib.copy(t, u) u={}; for k,v in pairs(t) do u[k]=v end; return u end
122
123 function lib.top(t,n, u)
124     u={};for k,v in pairs(t) do if k>n then break end; push(u,v) end; return u;end
125
126 --- Return a table's keys (sorted).
127 function lib.keys(t,u)
128     u={}
129     for k,_ in pairs(t) do if tostring(k):sub(1,1)~="_" then lib.push(u,k) end end
130     return lib.sort(u) end
131
132 --- Binary chop (assumes sorted lists)
133 function lib.bchop(t,val,lt,lo,hi, mid)
134     lt = lt or function(x,y) return x < y end
135     lo,hi = lo or 1, hi or #t
136     while lo <= hi do
137         mid = (lo+hi) // 2
138         if lt(t[mid],val) then lo=mid+1 else hi= mid-1 end end
139     return math.min(lo,#t) end
140
141 -----
142 return lib

```