

```

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
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
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040

```

```

101 ---
102 clear
103
104 local fails=0
105 local function ok(test,msg)
106   print("", test and "PASS" or "FAIL",msg or "")
107   if not test then
108     fails = fails+1 ; if the and the.dump then assert(test,msg) end end end
109
110 local demo={}
111 function demo.copy(      t,u)
112   t={a={b={c=10},d={e=200}}, f=300}
113   u= lib.copy(t)
114   t.a.b.c= 20
115   print(u.a.b.c)
116   oo(t)
117   oo(u)
118   lib.dent(u) end
119
120 function demo.rnd()
121   oo(rnds{23.1111111}) end
122
123 function demo.collect()
124   local function aux(x,y) return x*y end
125   oo(lib.collect({10,20,30},aux)) end
126
127 function demo.ent()
128   local a,b = lib.ent{a=9,b=7}
129   print(a,b)
130   ok(ish(lib.ent{a=9,b=7}, .98886), "entropy") end
131
132 function demo.items()
133   for x in items{10,20,30} do print(x) end
134   local n=0
135   print(33)
136   for x in items(the.file) do n=n+1; if n<=5 then print(100); oo(x) end end end
137
138 function demo.powerset()
139   for _,x in pairs(powerset{10,20,30,40,50}) do oo(x) end end
140
141 function demo.many( t)
142   t={};for j = 1,1000 do t[#t+1] = j end
143   print(900,"+", o(many(t,10,900)))
144   print(1,100,o(many(t,10,1,100)))
145   print(300,700, o(many(t,10,300,700))) end
146
147 function demo.new()
148   dent(seen.new{"Name","Age","gender","Weight--"}) end
149
150 function demo.clone( i,t,best,rest, x)
151   i={rows={},cols=nil}
152   the.file = "../etc/data/aut093.csv"
153   bins=xplain(the.file)
154   for _,row in pairs(i.rows) do
155     x=row[col].at end end
156
157 local function qq(i,q)
158   print(q[1], fmt("%15s=%-8s best=%s/%s rest=%s/%s",
159     i.cols[q[2]].name, q[3],q[4],q[5],q[6],q[7])) end
160
161 function demo.nb1()
162   local i = nb1(the.file);
163   local acc, out = score(i); print(acc); map(out,function(q) qq(i,q) end end
164
165 function demo.nb2()
166   the.file = "../etc/data/diabetes.csv"
167   the.goal = "positive"
168   local i = nb2(the.file);
169   abcd(i.log,true) end
170
171 function demo.nb2a()
172   the.file = "../etc/data/diabetes.csv"
173   the.goal = "positive"
174   for _,bins in pairs{2,5,9} do
175     print(bins)
176     the.bins = bins
177     local i = nb2(the.file);
178     abcd(i.log,true) end end
179
180 function demo.bins( t)
181   local t,n = {},30
182   for j=1,n do push(t, {x=j, y=j<.6*n and 1 or j<.8*n and 2 or 3}) end
183   map(bins(t,20),oo) end
184
185 function demo.nb3()
186   the.file = "../etc/data/diabetes.csv"
187   the.goal = "positive"
188   the.bins = 16
189   local i = nb3(the.file);
190   abcd(i.log,true)
191   local acc, out = score(i); map(out,function(q) qq(i,q) end end
192
193 ---
194 fails = 0
195 local defaults=lib.copy(the)
196 local todos = defaults.todo = "all" and slots(demo) or {defaults.todo}
197 for _,todo in pairs(todos) do
198   the = lib.copy(defaults)
199   math.randomseed(the.seed or 10019)
200   if demo[todo] then demo[todo]() end end
201
202 for k,v in pairs(_ENV) do if not b4[k] then print("??",k,type(v)) end end
203 os.exit(fails)
204
205 ---

```

```

213 ---
214 ---
215 ---
216 ---
217 ---
218 local lib = require"lib"
219 local has2,has3,inc,inc2,sort = lib.has2,lib.has3,lib.inc,lib.inc2,lib.sort
220
221 local nb={}
222 function nb.new() return {
223   h={}, nh=0,e={}, n=0, wait=the.wait,
224   bests=0,rests=0,best={}, rest={},log=log or {}, cols={} end
225
226 function nb.classify(i,t,use)
227   local hi,out = -1
228   for h,val in pairs(i.h) do
229     local prior = ((i.h[h] or 0) + the.K)/(i.n + the.K*i.nh)
230     local l = prior
231     for col,x in pairs(t) do
232       if x ~= "?" and i.cols[col].indep then
233         l=l*(has3(i.e,col,x,h) + the.M*prior)/((i.h[h] or 0) + the.M) end end
234       if l>hi then hi,out=l,h end end
235     return out end
236
237 function nb.test(i,t)
238   if i.n > the.wait then push(i.log,{want=t[#t], got=nb.classify(i,t)}) end end
239
240 function nb.train(i,t)
241   local more, kl = false, t[#t]
242   for col,x in pairs(t) do
243     if x ~= "?" then
244       more = true
245       inc3(i.e, col, x, kl)
246       if col ~= #t then
247         inc2(kl==the.goal and i.best or i.rest, col,x) end end end
248     if more then
249       i.n = i.n + 1
250       if not i.h[kl] then i.nh = i.nh + 1 end
251       inc(i.h, kl)
252       if kl==the.goal then i.bests=i.bests+1 else i.rests=i.rests+1 end end end
253
254 function nb.score(i)
255   local acc,out=0,{}
256   for key,x in pairs(i.log) do if x.want==x.got then acc=acc+1/#i.log end end
257   for col,xns in pairs(i.best) do
258     for x,b in pairs(xns) do
259       local r = has2(i.rest,col,x)
260       local rl = r/i.rests
261       local bl = b/i.bests
262       push(out, {100*(bl^2/(bl+rl))/1, col,x,b,i.bests,r,i.rests}) end end
263   return acc, sort(out,down1) end
264
265 return function(data, log)
266   local i = nb.new()
267   for row in items(data) do
268     if #i.cols == 0
269     then i.cols=collect(row,function(j,s) return {name=s,indep=truej==#row} end)
270     else test(i,row); train(i,row) end end
271   return i end
272 ---
273 ---
274 ---
275 ---
276 ---
277 local R=require
278 local the,lib,ako, nbl = R"the",R"lib",R"ako", R"learn101"
279 local collect = lib.collect
280
281 return function(data, log)
282   local tmp,xnums = {}
283   local function discretize(c,x, col)
284     if x ~= "?" then
285       col = xnums[c]
286       if col then x=(x - col.lo) // ((col.hi - col.lo+1E-32) / the.bins) end end
287     return x end
288
289   local function xnum(c,name)
290     if ako.xnum(name) then return {lo=1E32, hi=-1E32} end end
291
292   local function train(c,x, col)
293     col = xnums[c]
294     if col and x ~= "?" then
295       col.hi = math.max(x, col.hi)
296       col.lo = math.min(x, col.lo) end
297     return x end
298
299   for row in items(data) do
300     push(tmp, row)
301     if xnums then collect(row, train)
302     else xnums = collect(row,xnum) end end
303   for j=2,#tmp do tmp[j] = collect(tmp[j], discretize) end
304   return nbl(tmp) end
305 ---
306 ---
307 ---
308 ---
309 ---
310 local R=require
311 local nbl,bin,lib = R"learn101", R"bin", R"lib"
312 local collect,push = lib.collect,lib.push
313
314 return function(data, log)
315   local tmp, xnums = {}
316   local function discretize(c,x, col)
317     if x ~= "?" then
318       col = xnums[c]
319       if col then
320         for _,one in pairs(col.bins) do
321           if one.lo <= x and x < one.hi then return one.id end end end end
322       return x end
323
324   local function xnum(c,name)
325     if ako.xnum(name) then return {name=name, xys={},bins={}} end end
326
327   local function train(c,x,row)
328     if xnums[c] and x ~= "?" then push(xnums[c].xys, {x=x,y= row[#row]}) end end
329
330   for row in items(data) do
331     push(tmp,row)
332     if xnums then collect(row, function(c,x) return train(c,x,row) end)
333     else xnums = collect(row,xnum) end end
334   for where,col in pairs(xnums) do
335     col.bins = bin.Xys(col.xys,where); print(col.name,#col.bins) end
336   for j=2,#tmp do tmp[j] = collect(tmp[j], discretize) end
337   return nbl(tmp) end
338

```

```

339 ---
340 ---
341 ---
342 ---
343 ---
344 local the=require"the"
345 local lib=require"lib"
346 local fmt,per,push,sort = lib.fmt, lib.per, lib.push, lib.sort
347
348 local bin={}
349 function bin.new(id,at,name,lo,hi,n,div)
350   {id=id,at=at,name=name,lo=lo,hi=hi,n=n,div=div} end
351
352 function bin.show(i,negative)
353   local x,lo,hi,big, s = i.name, i.lo, i.hi, math.huge
354   if negative then
355     if lo==hi then s=fmt("%s!=",x,lo)
356     elseif hi==big then s=fmt("%s<=",x,lo)
357     elseif lo==big then s=fmt("%s>=",x,hi)
358     else s=fmt("%s<=%s and %s>=%s",x,lo,x,hi) end
359   else
360     if lo==hi then s=fmt("%s==",x,lo)
361     elseif hi==big then s=fmt("%s>=",x,lo)
362     elseif lo==big then s=fmt("%s<=",x,hi)
363     else s=fmt("%s<=%s < %s",lo,x,hi) end end
364   return s end
365
366 function bin.select(i,row)
367   local x, lo, hi = row[i.at], i.lo, i.hi
368   return x=="?" or lo == hi and lo == x or lo <= x and x < hi end
369 ---
370 ---
371 ---
372 ---
373 function bin.Merges(bins)
374   local j,n,new = 0,length(bins),{}
375   while j <= n do
376     j=j+1
377     a=bins[j]
378     if j < n then
379       b = bins[j+1]
380       if a.hi == b.lo then
381         a.hi = b.hi
382         a.div = (a.div*a.n + b.div*b.n)/(a.n+b.n)
383         a.n = a.n + b.n
384         j = j + 1 end end
385       push(new,a) end
386   return #new < #bins and bin.Merges(new) or bins end
387
388 local argmin
389 function bin.Xys(xys,at,name)
390   xys = sort(xys, upx)
391   local triviallySmall = the.cohen*(per(xys,.9).x - per(xys, .1).x)/2.56
392   local enoughItems = #xys / the.bins
393   local out = {}
394   argmin(1,xys, xys, triviallySmall, enoughItems, -math.huge, at.name, out)
395   out[#out].hi = math.huge
396   return out end
397
398 function argmin(lo, hi, xys, triviallySmall, enoughItems, b4, at, name,out)
399   local function add(f,z) f[z] = (f[z] or 0) + 1 end
400   local function sub(f,z) f[z] = f[z] - 1 end
401   local lhs, rhs, cut, div, xpect, xy = {},{}
402   for j=lo,hi do add(rhs, xys[j].y) end
403   div = ent(rhs)
404   if hi-lo+1 > 2*enoughItems then
405     for j=lo,hi - enoughItems do
406       add(lhs, xys[j].y)
407       sub(rhs, xys[j].y)
408       local n1,n2 = j - lo +1, hi-j
409       if n1 > enoughItems and n2 > enoughItems and
410         xys[j].x ~ xys[j+1].x and -- there is a break here
411         xys[j].x - xys[lo].x > triviallySmall and
412         xys[hi].x - xys[j].x > triviallySmall
413       then xpect = (n1*ent(lhs) + n2*ent(rhs)) / (n1+n2)
414         if xpect < div then -- cutting here simplifies things
415           cut, div = j, xpect end end end
416       end -- end if
417       if cut
418       then b4 = argmin(lo, cut, xys,triviallySmall,enoughItems,b4,at,name,out)
419       else -- if no cut then the original div was never updates and is still correct
420         b4 = argmin(cut+1,hi , xys,triviallySmall,enoughItems,b4,at,name,out)
421       end
422       b4 = push(out, bin.new(#out+1,at,name,b4,xys[hi].x, hi-lo+1,div)).hi end
423   return b4 end
424
425 return bin
426 ---
427 ---
428 ---
429 ---
430 ---
431 local lib=require"lib"
432 local bin=require"bin"
433 local map,push,sort = lib.map, lib.push, lib.sort
434
435 local rule={}
436 function rule.new(bins, t)
437   t = {}
438   for key,one in pairs(bins) do
439     t[one.at]=t[one.at] or{}; push(t[one.at],one) end
440   return {bins=t} end
441
442 function rule.selects(i,row)
443   local function ors(bins)
444     for key,x in pairs(bins) do if bin.select(x,row) then return true end end
445     return false end
446   for at,bins in pairs(i.bins) do if not ors(bins) then return false end end
447   return true end
448
449 function rule.show(i,bins)
450   local cat, order, ors
451   cat = function(t,sep) return table.concat(t,sep) end
452   order= function(a,b) return a.lo < b.lo end
453   ors= function(bins)
454     return cat(map(bin.Merges(sort(bins,order)),bin.show)," or ") end
455   return cat(map(i.bins, ors)," and ") end
456
457 return rule
458

```

```

459 ---
460 ---
461 ---
462 ---
463 ---
464 local ako={ }
465
466 ako.num = function(x) return x:find("[A-Z]" end
467 ako.goal = function(x) return x:find("[+]" end
468 ako.klass = function(x) return x:find("$" end
469 ako.ignore = function(x) return x:find("$" end
470 ako.weight = function(x) return x:find("-$" and -1 or 1 end
471 ako.xnum = function(x) return ako.num(x) and not ako.goal(x) end
472
473 return ako
474 ---
475 ---
476 ---
477 ---
478 local ako = require"ako"
479
480 local num = {}
481 function num.new(at,name)
482   (at=at or 0, name=name or "",
483    num=true, indep=false, n=0, w = ako.weight(name or ""),
484    lo=math.huge, hi=-math.huge, mu=0, m2=0, sd=0, bins={}) end
485
486 function num.add(i,x, d)
487   if x ~= "?" then
488     i.n = i.n+1
489     i.lo = math.min(x, i.lo)
490     i.hi = math.max(x, i.hi)
491     d = x - i.mu
492     i.mu = i.mu + d/i.n
493     i.m2 = i.m2 + d*(x - i.mu)
494     i.sd = ((i.m2<0 or i.n<2) and 0) or ((i.m2/(i.n - 1))^0.5) end
495   return x end
496
497 return num
498 ---
499 ---
500 ---
501 ---
502 ---
503 local sym = {}
504
505 function sym.new(at,name)
506   (at=at or 0, name=name or "",
507    num=false, indep=false, n=0,
508    has={}, most=0, mode=nil) end
509
510 function sym.add(i,x)
511   if x ~= "?" then
512     i.n = i.n + 1
513     i.has[x] = 1 + (i.has[x] or 0)
514     if i.has[x] > i.most then
515       i.mode,i.most = x,i.has[x] end end
516   return x end
517
518 return sym
519 ---
520 ---
521 ---
522 ---
523 local R=require
524 local ako,lib,sym,num = R"ako",R"lib",R"sym",R"num"
525 local norm,push = lib.norm, lib.push
526
527 local seen = {}
528 function seen.new(names)
529   return seen.init({names=names, klass=nil,xy= {}, x= {}, y={}},names) end
530
531 function seen.init(i, names)
532   for at,name in pairs(names) do
533     local now = (ako.num(name) and num.new or sym.new) (at,name)
534     push(i.xy, now)
535     if not ako.ignore(name) then
536       if not ako.goal(name) then now.indep = true end
537       if ako.klass(name) then i.klass=now end
538       push(now.indep and i.x or i.y, now) end end
539   return i end
540
541 function seen.add(i,row)
542   for _,col in pairs(i.xy) do
543     (col.nump and num or sym).add(col, row[col.at]) end
544   return row end
545
546 function seen.better(i,row1,row2)
547   local s1, s2, n, e = 0, 0, #i.y, math.exp(1)
548   for _,col in pairs(i.y) do
549     local a = norm(col.lo, col.hi, row1[col.at] )
550     local b = norm(col.lo, col.hi, row2[col.at] )
551     s1 = s1 - e*(col.w * (a - b) / n)
552     s2 = s2 - e*(col.w * (b - a) / n) end
553   return s1 / n < s2 / n end
554
555 return seen
556

```

```

557 ---
558 ---
559 ---
560 ---
561 ---
562 local seen = require"seen"
563 local lib = require"lib"
564 local map,sort,many = lib.map,lib.sort,lib.many
565 local items,slice = lib.items,lib.slice
566
567 ---
568 ---
569 ---
570 local eggs={}
571 function eggs.new() return {rows={}, cols={}} end
572
573 function eggs.Init(data, i)
574   i= eggs.new()
575   for row in items(data) do
576     if #i.cols==0 then i.cols=seen.new(row) else
577       push(i.rows, seen.add(i.cols,row)) end end
578   return i end
579
580 ---
581 ---
582 ---
583 ---
584 function eggs.mid(i,cols)
585   local function mid(col) return col.nump and col.mu or col.mode end
586   return map(cols or i.cols.y, mid) end
587
588 function eggs.div(i,cols)
589   local function div(col) return col.nump and col.sd or ent(col.has) end
590   return map(cols or i.cols.y, div) end
591
592 function eggs.clone(old,rows)
593   local i={rows={}, cols=seen.new(old.cols.names)}
594   for key,row in pairs(rows or {}) do seen.add(i.cols,row) end
595   return i end
596
597 ---
598 ---
599 ---
600 function eggs.dist(i,row1,row2)
601   local function sym(c,x,y) return x==y and 0 or 1 end
602   local function num(c,x,y)
603     if x=="?" then y = norm(c.lo, c.hi, y); x=y<.5 and 1 or 0
604     elseif y=="?" then x = norm(c.lo, c.hi, x); y=x<.5 and 1 or 0
605     else x,y = norm(c.lo, c.hi, x), norm(c.lo, c.hi, y) end
606     return math.abs(x-y) end
607   local function dist(c,x,y)
608     return x=="?" and y=="?" and 1 or (c.nump and num or sym) (c,x,y) end
609   local d, n = 0, #i.cols.x
610   for key,c in pairs(i.cols.x) do d=d+dist(c, row1[c.at], row2[c.at])^the.e end
611   return (d/n)^(1/the.e) end
612
613 ---
614 ---
615 ---
616 function eggs.bestRest(i)
617   i.rows = sort(i.rows, function(a,b) return seen.better(i.cols,a,b) end)
618   local n = (#i.rows)^the.best
619   return slice(i.rows, 1, n), -- top n things
620     many( i.rows, n*the.rest, n+1) end -- some sample of the rest
621
622 function eggs.Contrasts(i, rows1, rows2)
623   local function contrast(col)
624     local function asBin(x,ys, n,div)
625       n,div = ent(ys)
626       return bin.new(id, col.at, col.name, x, x, n, div) end
627     local symbols, xys, x = {},{}
628     for klass,rows in pairs{rows1,rows2} do
629       for key,row in pairs(rows) do
630         x = row[col.at]
631         if x ~= "?" then
632           if not col.nump then inc2(symbols,x,klass) end
633           push(xys, {x=x, y=class}) end end end
634     return col.nump and bins(xys, col.at) or collect(symbols, asBin) end
635   local out, tmp = {}
636   for key,col in pairs(i.cols.x) do
637     tmp = contrast(col)
638     if #tmp > 1 then
639       for key,one in pairs(tmp) do push(out, one) end end end
640   return out end
641
642 function eggs.xplain(i)
643   best, rest = eggs.bestRest(i)
644   return eggs.contrasts(i, best,rest) end
645
646 return eggs
647

```

```

648 ---
649 ---
650 ---
651 ---
652 ---
653 local lib=require"lib"
654
655 local abcd={}
656
657 function abcd.new(data,rx)
658   {data= data or "data",rx= rx or "rx",
659     known={},a={},b={},c={},d={},yes=0,no=0} end
660
661 function abcd.exists(i,x, new)
662   new = not i.known[x]
663   lib.inc(i.known,x)
664   if new then
665     i.a[x]=i.yes + i.no; i.b[x]=0; i.c[x]=0; i.d[x]=0 end end
666
667 function abcd.report(i, p,out,a,b,c,d,pd,pf,pn,f,acc,g,prec)
668   p = function (z) return math.floor(100*z + 0.5) end
669   out= {}
670   for x,xx in pairs( i.known ) do
671     pd,pf,pn,prec,g,f,acc = 0,0,0,0,0,0
672     a= (i.a[x] or 0); b= (i.b[x] or 0); c= (i.c[x] or 0); d= (i.d[x] or 0);
673     if b+d > 0 then pd = d / (b+d) end
674     if a+c > 0 then pf = c / (a+c) end
675     if a+c > 0 then pn = (b+d) / (a+c) end
676     if c+d > 0 then prec = d / (c+d) end
677     if 1-pf+pd > 0 then g=2*(1-pf) * pd / (1-pf+pd) end
678     if prec+pd > 0 then f=2*prec*pd / (prec + pd) end
679     if i.yes + i.no > 0 then
680       acc= i.yes / (i.yes + i.no) end
681     out[x] = {data=i.data,rx=i.rx,num=i.yes+i.no,a=a,b=b,c=c,d=d,acc=p(acc),
682               prec=p(prec), pd=p(pd), pf=p(pf),f=p(f), g=p(g), class=x} end
683   return out end
684
685 function abcd.pretty(t)
686   print" "
687   local s1 = "%10s|%10s|%4s|%4s|%4s|%4s"
688   local s2 = "%3s|%3s|%3s|%4s|%3s|%3s|"
689   local d,s = "----", (s1 .. s2)
690   print(fmt(s,"db","rx","a","b","c","d","acc","pd","pf","prec","f","g"))
691   print(fmt(s,d,d,d,d,d,d,d,d,d,d,d,d))
692   for key,x in pairs(lib.slots(t)) do
693     local u = t[x]
694     print(lib.fmt(s.." %s", u.data,u.rx,u.a, u.b, u.c, u.d,
695                  u.acc, u.pd, u.pf, u.prec, u.f, u.g, x)) end end
696
697 function abcd.adds(gotwants, show,data, rx)
698   local i = abcd.is(data,rx)
699   for key,one in pairs(gotwants) do
700     abcd.exists(i,one.want)
701     abcd.exists(i,one.got)
702     if one.want == one.got then i.yes=i.yes+1 else i.no=i.no+1 end
703     for x,xx in pairs(i.known) do
704       if one.want == x
705         then lib.inc(one.want == one.got and i.d or i.b, x)
706         else lib.inc(one.got == x and i.c or i.a, x) end end end
707   return show and abcd.pretty(abcd.report(i)) or abcd.report(i) end
708
709 return abcd.adds
710

```

```

711 ---
712 ---
713 ---
714 ---
715 ---
716 local _={}
717
718 ---
719 ---
720 ---
721 function _per(t,p) return t[ (p or .5)*#t//1 ] end
722
723 function _ent(t)
724   local n=0; for _,m in pairs(t) do n = n+m end
725   local e=0; for _,m in pairs(t) do if m>0 then e= e+m/n*math.log(m/n,2) end end
726   return -e,n end
727
728 function _norm(lo,hi,x) return math.abs(hi-lo)<1E-9 and 0 or (x-lo)/(hi-lo) end
729
730 ---
731 ---
732 ---
733 function _ish(x,y,z) return math.abs(x-y) <= (z or 0.001) end
734
735 ---
736 ---
737 ---
738
739 function _inc(f,a,n) f=f or{};f[a]=(f[a] or 0) + (n or 1) return f end
740 function _inc2(f,a,b,n) f=f or{};f[a]=_inc(f[a] or {},b,n); return f end
741 function _inc3(f,a,b,c,n) f=f or{};f[a]=_inc2(f[a] or {},b,c,n);return f end
742
743 function _has(f,a) return f[a] or 0 end
744 function _has2(f,a,b) return f[a] and _has(f[a],b) or 0 end
745 function _has3(f,a,b,c) return f[a] and _has2(f[a],b,c) or 0 end
746
747 ---
748 ---
749 ---
750 _unpack = table.unpack
751
752 function _push(t,x) t[1 + #t] = x; return x end
753
754 function _powerset(s)
755   local function aux(s)
756     local t = {}
757     for i = 1, #s do
758       for j = 1, #t do
759         t[#t+1] = {s[i], _unpack(t[j])} end end
760     return t end
761   return _sort(aux(s), function(a,b) return #a < #b end) end
762
763 ---
764 ---
765 ---
766
767 function _map(t, f, u)
768   u={}; for k,v in pairs(t) do u[1+#u]=f(v) end; return u end
769 function _collect(t,f,u)
770   u={}; for k,v in pairs(t) do u[k]=f(k,v) end; return u end
771 function _copy(t, u)
772   if type(t) ~= "table" then return t end
773   u={}; for k,v in pairs(t) do u[_copy(k)] = _copy(v) end; return u end
774
775 ---
776 ---
777 ---
778
779 function _sort(t,f) table.sort(t,f); return t end
780
781 function _upx(a,b) return a.x < b.x end
782 function _upl(a,b) return a[1] < b[1] end
783 function _downl(a,b) return a[1] > b[1] end
784
785 function _slots(t, u)
786   local function public(k) return tostring(k):sub(1,1) ~= "-" end
787   u={};for k,v in pairs(t) do if public(k) then u[1+#u]=k end end
788   return _sort(u) end
789
790 ---
791 ---
792 ---
793 function _any(a,lo,hi)
794   lo,hi = lo or 1, hi or #a; return a[ (lo+(hi-lo)*math.random())//1 ] end
795
796 function _many(a,n,lo,hi, u)
797   u={}; for j=1,n do _push(u, _any(a,lo,hi)) end; return u end
798
799 function _slice(a,lo,hi, u)
800   u,lo,hi = {},lo or 1,hi or #a; for j=lo,hi do u[1+#u]=a[j] end; return u end

```

```

801
802 --- string '2' thing
803 ---
804 ---
805
806 function _.words(s, sep, t)
807   sep="([^\s]+)"
808   t={}; for y in s:gmatch(sep) do t[1+#t] = y end; return t end
809
810 function _.things(s) return _.map(_.words(s), thing) end
811
812 function _.thing(x)
813   x = x:gmatch("%s*")
814   if x=="true" then return true elseif x=="false" then return false end
815   return tonumber(x) or x end
816
817 function _.items(src, f)
818   local function file()
819     src, f = io.input(src), f or _.things
820     return function() x=io.read()
821       print(6000, f)
822     end
823   end
824   local function tbl(x)
825     x, f = 0, f or function(z) return z end
826     return function() if x< #src then x=x+1; return f(src[x]) end end end
827   if src then
828     return type(src) == "string" and file() or tbl() end end
829
830 --- things '2' string
831 ---
832 ---
833
834 _.fmt = string.format
835
836 function _.oo(t) print(_.o(t)) end
837
838 function _.o(t, seen, u)
839   if type(t)~="table" then return tostring(t) end
840   seen = seen or {}
841   if seen[t] then return "..." end
842   seen[t] = t
843   local function show1(x) return _.o(x, seen) end
844   local function show2(k) return _.fmt("%s %s", k, _.o(t[k], seen)) end
845   u = #t>0 and _.map(t, show1) or _.map(_.slots(t), show2)
846   return (t._is or "")..{"..table.concat(u, " ")} end
847
848 function _.dent(t, seen, pre)
849   pre, seen = pre or "", seen or {}
850   if seen[t] then t = "..." end
851   if type(t)~="table" then return print(pre .. tostring(t)) end
852   seen[t]=t
853   for _, k in pairs(_.slots(t)) do
854     local v = t[k]
855     local after = type(v)=="table" and "\n" or "\t"
856     io.write(pre, ".", k, after)
857     if type(v)=="table"
858     then _.dent(v, seen, "| " .. pre)
859     else print(v) end end end
860
861 function _.rnds(t, f)
862   return _.map(t, function(x) return _.rnd(x, f) end) end
863
864 function _.rnd(x, f)
865   return _.fmt(type(x)=="number" and (x~x//1 and f or "%.2f") or "%s", x) end
866
867 return _

```