


```

156 -----
157 local Some=class("Some")
158 function Some:new()
159     self.kept, self.ok, self.n = {}, false, 0 end
160
161 function Some:add(x, a)
162     self.n = 1 + self.n
163     a = self.kept
164     if #a < the.keep then self.ok=false; push(a,x)
165     elseif r() < the.keep/self.n then self.ok=false; a[r(#a)]=x end end
166
167 function Some:has()
168     if not self.ok then table.sort(self.kept) end
169     self.ok = true
170     return self.kept end
171
172 -----
173 local Num=class("Num")
174 function Num:new(at,name)
175     self.at, self.name = at or 0, name or ""
176     self.w = self.name:find"$-" and -1 or 1
177     self.some=Some()
178     self.n,self.mu,self.m2,self.sd,self.lo,self.hi = 0,0,0,0,1E32,-1E32 end
179
180 function Num:add(x,_, a,d)
181     if x ~= "?" then
182         self.some:add(x)
183         self.n = self.n + 1
184         self.lo = min(x, self.lo)
185         self.hi = max(x, self.hi)
186         d = x - self.mu
187         self.mu = self.mu + d/self.n
188         self.m2 = self.m2 + d*(x - self.mu)
189         self.sd = (self.m2<0 or self.n<2) and 0 or ((self.m2/(self.n - 1))^0.5) end
190     return x end
191
192 function Num:mid() return self.mu end
193 function Num:div() return self.sd end
194
195 function Num:like(x,_)
196     local z, e, pi = 1E-64, math.exp(1), math.pi
197     if x < self.mu - 4*self.sd then return 0 end
198     if x > self.mu + 4*self.sd then return 0 end
199     return e^(-(x - self.mu)^2 / (z + 2*self.sd^2))/(z + (pi*2*self.sd^2)^.5) end
200
201 function Num:norm(x, lo,hi)
202     lo,hi = self.lo, self.hi
203     return x=="?" and x or hi-lo < 1E-9 and 0 or (x - lo)/(hi - lo) end
204
205 -----
206 local Sym=class("Sym")
207 function Sym:new(at,name)
208     self.at, self.name = at or 0, name or ""
209     self.n, self.has, self.mode, self.most = 0, {}, nil, 0 end
210
211 function Sym:add(x,inc)
212     if x ~= "?" then
213         inc = inc or 1
214         self.n = self.n + inc
215         self.has[x] = inc + (self.has[x] or 0)
216         if self.has[x] > self.most then self.most,self.mode = self.has[x],x end end
217     return x end
218
219 function Sym:mid() return self.mode end
220 function Sym:div() return ent(self.has) end
221
222 function Sym:like(x,prior)
223     return ((self.has[x] or 0) + the.m*prior)/(self.n + the.m) end
224
225 -----
226 local Cols=class("Cols")
227 function Cols:new(names, col)
228     self.names = names
229     self.all, self.x, self.y = {}, {}, {}
230     for at,name in pairs(names) do
231         col = push(self.all, (name:find"^[A-Z]" and Num or Sym) (at,name))
232         if not name:find"$" then
233             if name:find"$" then self.klass=col end
234             col.indep = not name:find"[+]"$"
235             push(col.indep and self.x or self.y, col) end end end
236
237 -----
238 local Egs=class("Egs")
239 function Egs:new() self.rows, self.cols = {}, nil end
240
241 function Egs:add(row, add)
242     add = function(col) col:add(row[col.at]) end
243     if self.cols then push(self.rows, map(self.cols,add)) else
244         self.cols = Cols(row) end end
245
246 function Egs:mid(cols)
247     return map(cols or self.cols.y, function(col) return col:mid() end) end
248
249 function Egs:div(cols)
250     return map(cols or self.cols.y, function(col) return col:div() end) end
251
252 function Egs:like(row,egs, n,prior,like,col)
253     n=0; for _,eg in pairs(egs) do n = n + #eg.rows end
254     prior = (#self.rows + the.k) / (n + the.k * #egs)
255     like = log(prior)
256     for at,x in pairs(row) do
257         col = self.cols.all[at]
258         if x ~= "?" and col.indep then like= like + log(col:like(x,prior)) end end
259     return like end
260
261 function Egs:better(row1,row2)
262     local s1, s2, n, e = 0, 0, #self.cols.y, math.exp(1)
263     for _,col in pairs(self.cols.y) do
264         local a = col:norm(row1[col.at])
265         local b = col:norm(row2[col.at])
266         s1 = s1 - e^(col.w * (a - b) / n)
267         s2 = s2 - e^(col.w * (b - a) / n) end
268     return s1 / n < s2 / n end
269
270 function Egs:betters()
271     return sort(self.rows, function(a,b) return self:better(a,b) end) end

```

```

271 -----
272 function go.the() ooo(the) end
273
274 function go.ent() ok(abs(1.3788 - ent(a=4,b=2,c=1)) < 0.001,"enting") end
275
276 function go.ooo() ooo{cc=1,bb={ff=4,dd=5,bb=6}, aa=3} end
277
278 function go.copy( t,u)
279     t = {a=1,b=2,c={d=3,e=4,f={g=5,h=6}}}
280     u = copy(t)
281     t.c.f.g = 100
282     ok(u.c.f.g ~= t.c.f.g, "deep copy") end
283
284 function go.rnds() ooo(rnds{3.421212, 10.1121, 9.1111, 3.44444}) end
285
286 function go.csv( n)
287     n=0; for row in csv(the.file) do n=n+1 end; ok(n==399,"stuff") end
288
289 function go.some( s)
290     the.keep = 64
291     s = Some(); for i=1,10^6 do s:add(i) end
292     ooo(s:has()) end
293
294 function go.num( n,mu,sd)
295     n, mu, sd = Num(), 10, 1
296     for i=1,10^3 do
297         n:add(mu + sd*math.sqrt(-2*math.log(r()))*math.cos(2*math.pi*r())) end
298     ok(abs(n:mid() - mu) < 0.025, "sd")
299     ok(abs(n:div() - sd) < 0.05, "div") end
300
301 function go.sym( s,mu,sd)
302     s= Sym()
303     for i=1,100 do
304         for k,n in pairs{a=4,b=2,c=1} do s:add(k,n) end end
305     ooo(s:has) end
306
307 -----
308 the = settings(the,help)
309
310 if pcall(debug.getlocal, 4, 1)
311 then return {Num=Num, Sym=Sym, Egs=Egs} -- called as sub-module. return classes
312 else the = cli(the) -- update 'the' from command line
313     demos(the,go) -- run some demos
314     for k,v in pairs(_ENV) do if not b4[k] then print("?",k,type(v)) end end
315     os.exit(fails) end

```