

```

1  -----
2  ---
3  ---
4  ---
5  ---
6  ---
7  ---
8  ---
9  ---
10 -----
11
12 local help=[[
13 bore == best or rest
14 (c) 2022, Tim Menzies, BSD 2-clause license.
15
16 USAGE:
17   lua bore.lua [OPTIONS]
18
19 OPTIONS:
20   -Dump          stack dump on error = false
21   -Format S      format string          = $5.2f
22   -best F        best space              = .15
23   -cohen F       Cohen's delta           = .35
24   -data N        data file               = etc/data/auto93.csv
25   -furthest F    far                    = .9
26   -help          show help               = false
27   -seed I        random seed             = 10019
28   -todo S        start-up action         = nothing
29 ]]
30 -----
31 ---
32 ---
33 ---
34 ---
35 ---
36 ---
37
38 local b4={}; for k, _ in pairs(_ENV) do b4[k]=k end
39 local big = 1E32
40 local tiny = 1E-32
41 local the = {}
42
43 local function atom(x)
44   if type(x) ~= "string" then return x end
45   x = x:match("%s*(~)%s*$")
46   if x == "true" then return true elseif x == "false" then return false end
47   return tonumber(x) or x end
48
49 local function atoms(x, t)
50   t = {}; for y in x:gmatch(sep or "([^\s]+)") do t[1+#t]=atom(y) end; return t end
51
52 local function cli(txt, t)
53   t = {}
54   txt:gsub("\n [~]([^\s+)]*\n)%s*([^\s+)]*", function(key, x)
55     for n, flag in ipairs(flag) do
56       if flag:sub(1,1) == "-" and key:find("^"..flag:sub(2)..".*") then
57         x = x == "false" and true or x == "true" and "false" or arg[n+1] end end
58       t[key] = atom(x) end)
59   return t end
60
61 local fmt = string.format
62
63 local function sort(t, f) table.sort(t, f); return t end
64
65 local function slots(t, u)
66   u = {}; for k, v in pairs(t) do l=tostring(k); if l:sub(1,1) ~= "_" then u[1+#u]=k end end;
67   return sort(u) end
68
69 local function main(the, help, demos)
70   if the.help then print(help) else
71     for _, todo in pairs(the.todo == "all" and slots(demos) or {the.todo}) do
72       math.randomseed(the.seed)
73       if type(demos[todo]) == "function" then demos[todo]() end end end
74   os.exit(demos.fail) end
75
76 local function map(t, f, u)
77   u = {}; for k, v in pairs(t) do u[1+#u]=f(v) end; return u end
78
79 local function tablep(t) return type(t) == "table" end
80
81 local function o(t, seen)
82   seen = seen or {}
83   if not tablep(t) then return tostring(t) end
84   if seen[t] then return "..." end
85   seen[t] = t
86   local key = function(k) return fmt("%s %s", k, o(t[k], seen)) end
87   local u = #t > 0 and map(t, function(x) return o(x, seen) end) or map(slots(t), key)
88   return '{ ' .. table.concat(u, ", " .. " }" end
89
90 local function oo(t) print(o(t)) end
91
92 local function rows(file, x, prep)
93   file = io.input(file)
94   return function()
95     x = io.read(); if x then return atoms(x) else io.close(file) end end end
96
97 local function sum(t, f, n)
98   n = 0; for _, v in pairs(t) do n = n + f(v) end; return n end
99
100 local function tree(t, seen, pre, txt, v)
101   pre, seen = pre or "", seen or {}
102   if not tablep(t) then return print(fmt("%s %s", pre, t)) end
103   if seen[t] then return print(fmt("%s...", pre)) end
104   seen[t] = t
105   for _, k in pairs(slots(t)) do
106     v = t[k]
107     if tablep(v)
108       then print(fmt("%s %s", pre, k)); tree(v, seen, pre .. " " )
109     else print(fmt("%s %s = %s", pre, k, v)) end end end

```

```

100 -----
101 ---
102 ---
103 ---
104 ---
105 ---
106 ---
107 ---
108 ---
109 -----
110
111 ---
112 ---
113 ---
114 ---
115 ---
116 local as = setmetatable
117 local function obj(t)
118   t = {__tostring=o; t.__index=t
119   return as(t, {__call=function(_, ...) return t.new(_, ...) end}) end
120
121 ---
122 ---
123 ---
124
125 local function col(at, txt, i)
126   i = {n=0, at=at or 0, txt=txt or "", has={}}
127   i.w = i.txt:find"$" and -1 or 1
128   return i end
129
130 local function add(self, x, inc)
131   if x ~= "" then
132     inc = inc or 1
133     self.n = self.n + inc
134     self:add(x, inc) end
135   return self end
136
137 ---
138 ---
139 ---
140
141 local Num = obj{}
142 function Num:new(at, x, new)
143   new = as(col(at, x), Num)
144   new.mu, new.m2, new.lo, new.hi = 0, 0, big, -big
145   return new end
146
147 function Num:add(x, d)
148   d = x - self.mu
149   self.mu = self.mu + d/self.n
150   self.m2 = self.m2 + d*(x - self.mu)
151   self.sd = (self.n < 2 or self.m2 < 0) and 0 or (self.m2/(self.n-1))^5
152   if x > self.hi then self.hi = x end
153   if x < self.lo then self.lo = x end end
154
155 function Num:norm(x)
156   return self.hi - self.lo < tiny and 0 or (x - self.lo)/(self.hi - self.lo) end
157
158 function Num:heaven(x, heaven)
159   return ((self.w > 0 and 1 or 0) - self:norm(x))^the.p end
160
161 ---
162 ---
163 ---
164
165 local Sym = obj{}
166 function Sym:new(at, x, inc, new)
167   new = as(col(at, x), Sym); new.most = 0; return new end
168
169 function Sym:add(x, inc)
170   self.has[x] = inc + (self.has[x] or 0)
171   if self.has[x] > self.most then self.most, self.mode = self.has[x], x end end
172
173 function Sym:div()
174   local function plopp(n, p) p = n/self.n; return p*math.log(p, 2) end
175   return -sum(self.has, plopp) end
176
177 ---
178 ---
179 ---
180
181 local Skip = obj{}
182 function Skip:new(at, x) return as(col(at, x), Skip) end
183 function Skip:add(x, inc) return x end
184
185 ---
186 ---
187 ---
188
189 local Cols = obj{}
190 function Cols:new(headers, self, col, here)
191   self = as({all={}, x={}, y={},}, Cols)
192   for at, x in pairs(headers) do
193     if x:find"$" then self.all[at] = Skip(at, x) else
194       col = (x:find"[A-Z]" and Num or Sym) (at, x)
195       self.all[at] = col
196       here = x:find"[+-]" and self.y or self.x
197       here[1+#here] = col end end
198   return self end
199
200 function Cols:add(t)
201   for _, col in pairs(self.all) do col:add(t[col.at]) end
202   return t end
203
204 function Cols:clone(rows, new)
205   new = new or Cols(map(self.cols.all, function(x) return x.txt end))
206   for _, row in pairs(rows or {}) do new:add(row) end
207   return {rows=rows, cols=new} end
208
209 ---
210 ---
211 ---
212
213 local Data = obj{}
214 function Data:new(inits, new)
215   new = as({rows={}, heavens=Num(),}, Data)
216   if type(inits) == "string" then for row in csv(inits) do new:add(row) end end
217   if type(inits) == "table" then for _, row in pairs(inits) do new:add(row) end end
218   return new end
219
220 function Data:add(t, n)
221   if self.cols then self:addData(t) else
222     self.cols = Cols(t)
223     self.best = self.cols:clone()
224     self.rest = self.cols:clone() end end
225
226 function Data:addData(t, n)
227   self.rows[1+#self.rows] = self.cols:add(t)
228   n = self.heavens.norm( self.heavens.add(self.heaven(t)))
229   (n >= the.best and self.best or self.rest):add(t) end
230
231 function Data:heaven(t)
232   heaven = function(col) return col:heaven(t[col.at]) end
233   return (sum(self.cols.y, heaven)/#self.cols.y)^(1/the.p) end

```

```

228 -----
229
230
231
232
233
234
235 local Demos = {fails=0}
236
237 local function asserts(test, msg)
238     print(test and "PASS: " or "FAIL: ", msg or "")
239     if not test then
240         Demos.fails = Demos.fails+1
241         if the.Dump then assert(test, msg) end end end
242
243 function Demos.the()      oo(the) end
244 function Demos.col()      oo(col(10, "Mpg-")) end
245 function Demos.num(      n) n=Num();
246     for x=1,1000 do add(n,x) end; print(n) end
247
248 function Demos.sym(      s)
249     s=sym(); for _,x in pairs{1,1,1,1,2,2,3} do add(s,x) end
250     asserts(s:div() - 1.376 < 0.005, "entropy") end
251
252 function Demos.cols(      c)
253     print(Cols({{"Clnrs", "Weight", "Hp.", "Lbs-",
254         "Acc+", "Model", "origin", "Mpg+"}}))
255     end
256
257 the = cli(help)
258 main(the, help, Demos)

```