

Aug 27, 22 9:35

csv.lua

Page 1/3

```

1 local b4={}; for k,v in pairs(_ENV) do b4[k]=v end -- LUA trivia. Ignore.
2 local help={
3   CSV : summarized csv file
4   (c) 2022 Tim Menzies <tim@ieee.org> BSD-2 license
5
6   USAGE: lua seen.lua [OPTIONS]
7
8   OPTIONS:
9   -e --eg      start-up example      = nothing
10  -d --dump     on test failure, exit with stack dump = false
11  -f --file     file with csv data     = ../data/auto93.csv
12  -h --help     show help              = false
13  -n --nums     number of nums to keep = 512
14  -s --seed     random number seed    = 10019
15  -S --separator feild separator      = ,
16
17 -- ## Misc routines
18 -- ### Handle Settings
19 -- Parse 'the' config settings from 'help'.
20 local the={}
21 local function coerce(s)
22   local function coercesl(s1)
23     if s1=="true" then return true end
24     if s1=="false" then return false end
25     return sl end
26   return math.tointeger(s) or tonumber(s) or coercesl(s:match"^%s*(-)%s*$") end
27
28 help:gsub("[\n-][%S+]+[%S+][\n-][%S+]+[%S+]+[%S+]",
29   function(k,x) the[k]=coerce(x) end)
30
31 -- Update settings from values on command-line flags. Booleans need no values
32 -- (we just flip the defaults).
33 local function cli(t)
34   for slot,v in pairs(t) do
35     v = tostring(v)
36     for n,x in ipairs(arg) do
37       if x=="-"..(slot:sub(1,1)) or x=="-"..slot then
38         v = v=="false" and "true" or v=="true" and "false" or arg[n+1] end end
39     t[slot] = coerce(v) end
40   if t.help then os.exit(print("lua"..help.."")) end
41   return t end
42
43 -- ### Linting code
44 -- Find rogue locals.
45 local function rogues()
46   for k,v in pairs(_ENV) do if not b4[k] then print("?",k,type(v)) end end end
47
48 -- ### Strings
49 -- 'o' generates a string from a nested table.
50 local function o(t)
51   if type(t) ~= "table" then return tostring(t) end
52   local function show(k,v)
53     if not tostring(k):find"^_" then
54       v = o(v)
55       return #t==0 and string.format("%s%s",k,v) or tostring(v) end end
56   local u={}; for k,v in pairs(t) do u[1+#u] = show(k,v) end
57   if #t==0 then table.sort(u) end
58   return (t._is or "").."{"..table.concat(u,"").."}" end
59
60 -- 'oo' prints the string from 'o'.
61 local function oo(t) print(o(t)) return t end
62
63 -- ### Lists
64 -- Deepcopy
65 local function copy(t)
66   if type(t) ~= "table" then return t end
67   local u={}; for k,v in pairs(t) do u[k] = copy(v) end
68   return setmetatable(u, getmetatable(t)) end
69
70 -- Return the 'p'-th thing from the sorted list 't'.
71 local function per(t,p)
72   p=math.floor(((p or .5)*#t)+.5); return t[math.max(1,math.min(#t,p))] end
73
74 -- Add to 't', return 'x'.
75 local function push(t,x) t[1+#t]=x; return x end
76
77 -- ## Call 'fun' on each row. Row cells are divided in 'the.separator'.
78 local function csv(fname,fun)
79   local sep = ("["..the.separator.."]+")
80   local src = io.input(fname)
81   while true do
82     local s = io.read()
83     if not s then return io.close(src) else
84       local t={}
85       for sl in s:gmatch(sep) do t[1+#t] = coerce(sl) end
86       fun(t) end end end

```

Aug 27, 22 9:35

csv.lua

Page 2/3

```

87 -- -----
88 -- ## Objects
89 -- 'Data' is a holder of 'rows' and their summaries (in 'cols').
90 local function Data() return {_is = "Data",
91   cols = nil, -- summaries of data
92   rows = {} -- kept data
93 } end
94
95 -- 'Columns' Holds of summaries of columns.
96 -- Columns are created once, then may appear in multiple slots.
97 local function Cols() return {
98   _is = "Cols",
99   names={}, -- all column names
100   all={}, -- all the columns (including the skipped ones)
101   klass=nil, -- the single dependent klass column (if it exists)
102   x={}, -- independent columns (that are not skipped)
103   y={}, -- dependent columns (that are not skipped)
104 } end
105
106 -- 'Sym's summarize a stream of symbols.
107 local function Sym(c,s)
108   return {_is="Sym",
109     n=0, -- items seen
110     at=c or 0, -- column position
111     names=s or "", -- column name
112     _has={} -- kept data
113   } end
114
115 -- 'Num' ummarizes a stream of numbers.
116 local function Num(c,s)
117   return {_is="Nums",
118     n=0,at=c or 0, name=s or "", _has={}, -- as per Sym
119     isNum=true, -- mark that this is a number
120     lo=math.huge, -- lowest seen
121     hi=-math.huge, -- highest seen
122     isSorted=true, -- no updates since last sort of data
123     w = ((s or ""):find"$" and -1 or 1)
124   } end
125
126 -- 'Row' holds one record
127 local function Row(t) return {_is="Row",
128   cells=t, -- one record
129   cooked=copy(t), -- used if we discretize data
130   isEvald=false -- true if y-values evaluated.
131 } end
132
133 -- ## Data
134 -- Add one thing to 'col'. For Num, keep at most 'nums' items.
135 local function add(col,v)
136   if v=="?" then
137     col.n = col.n + 1
138     if not col.isNum then col._has[v] = 1 + (col._has[v] or 0) else
139       col.lo = math.min(v, col.lo)
140       col.hi = math.max(v, col.hi)
141     end
142     local pos
143     if #col._has < the.nums then pos = 1 + (#col._has)
144     elseif math.random() < the.nums/col.n then pos = math.random(#col._has) end
145     if pos then col.isSorted = false
146     col._has[pos] = tonumber(v) end end end end
147
148 local function adds(col,t) for _,x in pairs(t) do add(col,x) end; return col end
149
150 -- Add a 'row' to 'data'. Calls 'add()' to update the 'cols' with new values.
151 local function record(data,xs)
152   local row = push(data.rows, xs.cells and xs or Row(xs)) -- ensure xs is a Row
153   for _,col in pairs(data.cols.x, data.cols.y) do
154     add(col, row.cells[col.at]) end end end
155
156 -- Generate rows from some 'src'. If 'src' is a string, read rows from file;
157 -- else read rows from a 'src' table. When reading, use row1 to define columns.
158 local function records(src, data,head,body)
159   function head(sNames)
160     local cols = Cols()
161     cols.names = sNames
162     for c,s in pairs(sNames) do
163       local col = push(cols.all, (s:find"^[A-Z]" and Num or Sym)(c,s))
164       if not s:find"$" then -- some columns are skipped
165         push(s:find"[^|]+" and cols.y or cols.x, col) -- some cols are goal cols
166         if s:find"$" then cols.klass=col end end end
167     return cols
168   end
169   function body(t) -- treat first row differently (defines the columns)
170     if data.cols then record(data,t) else data.cols=head(t) end
171   end
172   data = Data()
173   if type(src)=="string" then csv(src, body) else
174     for _,t in pairs(src or {}) do body(t) end end
175   return data end
176
177 -- ## Query
178 -- Return kept numbers, sorted.
179 local function nums(num)
180   if not num.isSorted then table.sort(num._has); num.isSorted=true end
181   return num._has end
182
183 -- Diversity (standard deviation for Nums, entropy for Syms)
184 local function div(col)
185   if col.isNum then local a=nums(col); return (per(a,.9)-per(a,.1))/2.58 else
186     local function fun(p) return p*math.log(p,2) end
187     local e=0
188     for _,n in pairs(col._has) do if n>0 then e=e-fun(n/col.n) end end
189     return e end end
190
191 -- Central tendency (median for Nums, mode for Syms)
192 local function mid(col)
193   if col.isNum then return per(nums(col),.5) else
194     local most,mode = -1
195     for k,v in pairs(col._has) do if v>most then mode,most=k,v end end
196     return mode end end
197
198 -- Diversity (standard deviation for Nums, entropy for Syms)
199 local function div2(col)
200   if col.isNum then local a=nums(col); return (per(a,.9)-per(a,.1))/2.58 else
201     local function fun(p) return p*math.log(p,2) end
202     local e=0
203     for _,n in pairs(col._has) do if n>0 then e=e-fun(n/col.n) end end
204     return e end end
205
206 -- For 'showCols' (default='data.cols.x') in 'data', report 'fun' (default='mid').
207 local function stats(data, showCols,fun, t)
208   showCols, fun = showCols or data.cols.y, fun or mid
209   t={}; for _,col in pairs(showCols) do t[col.name]=fun(col) end; return t end
210

```

```

211
212 -- -----
213 -- ## Test Engine
214 local eg, fails = {},0
215
216 -- [1] reset random number seed before running something.
217 -- [2] Cache the defaults settings, and [3] restore them after the test
218 -- [4] Print error messages or stack dumps as required.
219 -- Return true if this all went well.
220 local function runs(k, old, status, out, msg)
221     if not eg[k] then return end
222     math.randomseed(the.seed) -- reset seed [1]
223     old={}; for k,v in pairs(the) do old[k]=v end -- [2]
224     if the.dump then
225         status,out = true, eg[k]()
226     else
227         status,out = pcall(eg[k]) -- pcall means we do not crash and dump on error
228     end
229     for k,v in pairs(old) do the[k]=v end -- restore old settings [3]
230     msg = status and ((out==true and "PASS") or "FAIL") or "CRASH" -- [4]
231     print("!!!!!!", msg, k, status)
232     return out or err end
233
234 -- -----
235 -- ## Tests
236 -- Test that the test happens when something crashes?
237 function eg.BAD() print(eg.dont.have.this.field) end
238
239 -- Sort all test names.
240 function eg.LIST( t )
241     t={}; for k,_ in pairs(eg) do t[1+#t]=k end; table.sort(t); return t end
242
243 -- List test names.
244 function eg.LS()
245     print("\nExamples lua csv -e...")
246     for _,k in pairs(eg.LIST()) do print(string.format("%s",k)) end
247     return true end
248
249 -- Run all tests
250 function eg.ALL()
251     for _,k in pairs(eg.LIST()) do
252         if k ~= "ALL" then
253             print("\n-----")
254             if not runs(k) then fails=fails+ 1 end end end
255     return true end
256
257 -- Settings come from big string top of "sam.lua"
258 -- (maybe updated from comamnd line)
259 function eg.the() oo(the); return true end
260
261 -- The middle and diversity of a set of symbols is called "mode"
262 -- and "entropy" (and the latter is zero when all the symbols
263 -- are the same).
264 function eg.sym( sym,entropy,mode)
265     sym= adds(Sym(), {"a","a","a","a","b","b","c"})
266     mode, entropy = mid(sym), div(sym)
267     entropy = (1000*entropy)//1/1000
268     oo({mid=mode, div=entropy})
269     return mode=="a" and 1.37 <= entropy and entropy <=1.38 end
270
271 -- The middle and diversity of a set of numbers is called "median"
272 -- and "standard deviation" (and the latter is zero when all the nums
273 -- are the same).
274 function eg.num( num)
275     num=Num()
276     for i=1,100 do add(num,i) end
277     local med,ent = mid(num), div(num)
278     print(mid(num),div(num))
279     return 50<= med and med<= 52 and 30.5 <ent and ent <32 end
280
281 -- Nums store only a sample of the numbers added to it (and that storage
282 -- is done such that the kept numbers span the range of inputs).
283 function eg.bignum( num)
284     num=Num()
285     the.nums = 32
286     for i=1,1000 do add(num,i) end
287     oo(nums(num))
288     return 32==#num._has; end
289
290 -- Show we can read csv files.
291 function eg.csv()
292     local n=0
293     csv("../data/aut093.csv",function(row)
294         n=n+1; if n> 10 then return else oo(row) end end); return true end
295
296 -- Print some stats on columns.
297 function eg.stats()
298     oo(stats(records("../data/aut093.csv"))); return true end
299
300 -- -----
301 the = cli(the)
302 runs(the.eg)
303 rogues()
304 os.exit(fails)

```