

Aug 27, 22 9:16

csv.lua

Page 1/3

```

1 local b4={}; for k,v in pairs(_ENV) do b4[k]=v end
2 local help={
3 SEEN : 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 coercel(s1)
23     if s1=="true" then return true end
24     if s1=="false" then return false end
25     return s1 end
26   return math.tointeger(s) or tonumber(s) or coercel(s:match"^%s*(-)%s*$") end
27
28 help:gsub("^\\n[\\-|[%S]+[%s]+[\\-|][%S]+[\\^n]+=[%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=="-." then slot=sub(1,1) or x=="-." 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("un"..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 9:16

csv.lua

Page 2/3

```

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

```

```

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

```