1	<u> </u>
2	/_ \
4	
5	
6 7	_\\\\\\\\\\\\\\\\\\\\\\\\\\\\
8	
9	cam lua . Somi-cuporviced And Multi-chiestive evalanation
11	sam.lua : Semi-supervised And Multi-objective explanation (c) 2022 Tim Menzies <timm@ieee.org> BSD-2 license</timm@ieee.org>
12	
13	In this code: Line strive to be 80 chars (or less)
15	Line strive to be 80 chars (or less) Two spaces before function argumnets denote optionals.
16 17	Four spaces before function argumnets denote local variables Private functions start with '_'
18	Arguments of private functions do anything at all
19	Arguments of private functions do anything at all Local variables inside functions do anything at all
20 21	Local variables inside functions do anything at all - Arguments of public functions use type hints - Variable 'x' is is anything - Prefix 'is' is a boolean - Prefix 't' is a boolean - Prefix 't' is a filename - Prefix 't' is a filename - Prefix 'n' is a string - Prefix 's' is a string - 'x' is anything (table or number of boolean or string) - 'y' is a simple value (number or boolean or string) - Suffix 's' is a list of things - Tables are 't' or, using the above, a table of numbers would be 'ns' - Type names are lower case versions of constuctors, so in this code, 'cols', 'data', 'num', 'sym' are made by functions 'Cols' 'Data', 'Num', 'Sym local =require'him'
22	Prefix 'is' is a boolean
23	Prefix 'fun' is a function
24 25	Prefix 'n' is a string
26	Prefix 's' is a string
27 28	Prefix 'c' is a column index
29	'x' is anything (table or number of boolean or string
30	'v' is a simple value (number or boolean or string)
31 32	Surrix 's' is a list of things Tables are 't' or, using the above, a table of numbers would be 'ns'
33	Type names are lower case versions of constuctors. so in this code,
34	'cols', 'data', 'num', 'sym' are made by functions 'Cols' 'Data', 'Num', 'Sym
35 36	<pre>local l=require"lib" local the=l.settings([[</pre>
37	local the=1.settings([[SAM : Semi-supervised And Multi-objective explainations
38 39	(c) 2022 Tim Menzies <timm@ieee.org> BSD-2 license</timm@ieee.org>
10	USAGE: lua eg.lua [OPTIONS]
11	OPTIONS:
12 13	-eeq start-up example = nothing
14	
45 46	-hhelp show help = false -nnums how many numbers to keep = 256 -pp distance coeffecient = 2 -sseed random number seed = 1001911
17	-sseed random number seed = 10019]])
48	
19 50	Commonly used lib functions. local o,oo,per,push = 1.o,1.oo,1.per, 1.push
51	
52 53	<pre>local Data, Cols, Sym, Num, Row local add, adds, clone, dist, div, mid, nums, record, read, stats</pre>
54	
55	Classes Holder of 'rows' and their sumamries (in 'cols').
56 57	function Data() return {cols=nil, rows={}} end
58	
59 50	Hoder of summaries function Cols() return {klass=nil,names={},nums={}, x={}, y={}, all={}} end
51	
52 53	Summary of a stream of symbols.
54	<pre>function Sym(c,s) return {n=0,at=c or 0, name=s or "", _has={}} end</pre>
55	
56 57	Summary of a stream of numbers. function Num(c,s)
58	return (n=0, at=c or 0, name=s or "", _has={},
59 70	<pre>return (n=0,at=c or 0, name=s or "", _has={}, isNum=true, lo= math.huge, hi= -math.huge, sorted=true, w=(s or ""):find"-\$" and -1 or 1 end</pre>
ro 71	
72	Hold one record, in 'cells' (and 'cooked' is for discretized data). function Row(t) return (cells=t, cooked=1.copy(t)) end
73 74	function Row(t) return {cells=t, cooked=1.copy(t)} end
75	Data Functions
76	Update Add one 'col'. For Num, keep at most 'nums' items.
77 78	function add(col,v)
19	<pre>function add(col,v) if v==""" ten col.n = col.n + 1</pre>
30 31	<pre>col.n = col.n + 1 if not col.isNum then colhas[v] = 1 + (colhas[v] or 0) else</pre>
32	col.lo = math.min(v, col.lo) col.hi = math.max(v, col.hi)
33	col.hi = math.max(v, col.hi)
34 35	<pre>local pos if #colhas < the.nums</pre>
36	elseif math.random() < the.nums/col.n then pos = math.random(#col.has) end if pos then col.sorted = false
37 38	<pre>if pos then col.sorted = false</pre>
39	
90 91	Add many items function adds(col,t) for _,v in pairs(t) do add(col,v) end; return col end
91 92	
93	Query Return kept numbers, sorted.
94 95	keturn kept numbers, sorted. function nums(num)
96	<pre>function nums(num) if not num.sorted then table.sort(numhas); num.sorted=true end</pre>
97	return numhas end
98 99	Diversity (standard deviation for Nums, entropy for Syms)
00	function div(col)
01 02	<pre>if col.isNum then local a=nums(col); return (per(a,.9)-per(a,.1))/2.58 else local function fun(p) return p*math.log(p,2) end</pre>
03	local e=0
04	<pre>for _,n in pairs(colhas) do if n>0 then e=e-fun(n/col.n) end end return o end end</pre>
05 06	return e end end
07	Central tendancy (median for Nums, mode for Syms)
08 09	<pre>function mid(col) if col.isNum then return per(nums(col),.5) else</pre>
10	<pre>local most, mode = -1</pre>
11	<pre>for k,v in pairs(colhas) do if v>most then mode,most=k,v end end</pre>
12 13	return mode end end
14	Data functions
15 16	Create Processes table of name strings (from row1 of csv file)
17	local cols = Cols()
18	<pre>local cols = Cols() cols.names = namess</pre>

```
for c,s in pairs(sNames) do
          ior c, s in pairs(sNames) do
local col = push(cols.all, -- Numerics start with Uppercase.
(s:find*"[A-Z]"* and Num or Sym((c,s))
if not s:find*"S* then -- some columns are skipped
push(s:find*["*-" and cols.y or cols.x, col) -- some cols are goal cols
if s:find*["$" then cols.klass=col end end end
       return cols end
    -- If 'src' is a string, read rows from file; else read rows from a 'src' table -- When reading, use rowl to define the column headers.
 function read(src, data, fun)
data = data or Data()
function fun(t) if data.cols then record(data,t) else data.cols=_head(t) end end
if type(src)=="string" then l.csv(src,fun)
else for _rt in pairs(src or {}) do fun(t) end end
      return data end
      - Return a new data with same structure as 'datal'. Optionally, oad in 'rows'.
 function clone (data1, rows)
data2=Data()
data2=Data()
data2.cols = _head(data1.cols.names)
for _,row in pairs(rows or {})) do record(data2,row) end
return data2 end
   --- -- Dupate

-- Add a new 'row' to 'data', updating the 'cols' with the new values.

function record(data,xs)

local row= push(data.rows, xs.cells and xs or Row(xs)) -- ensure xs is a Row

for _rtodo in pairs(data.cols.x, data.cols.y) do
          for _,col in pairs(todo) do
  add(col, row.cells[col.at]) end end end
  ---- Or 'showCols' (default='data.cols.x') in 'data', report 'fun' (default='mid').

function stats(data, showCols,fun, t)

showCols, fun = showCols or data.cols.y, fun or mid

t=(); for _,col in pairs(showCols) do t[col.name]=fun(col) end; return t end
 else v1,v2 = norm(v1), norm(v2) end
return math.abs(v1-v2) end
  -- Distance between two rows (returns 0..1)
function dist(data,tl,t2)
local d = 0
for _,col in pairs(data.cols.x) do
d = d + _distl(col, tl.cells[col.at], t2.cells[col.at])^the.p end
return (d/#data.cols.x)^(1/the.p) end
         That's all folks.
  return (the=the,
Data=Data, Cols=Cols, Sym=Sym, Num=Num, Row=Row,
add=add, adds=adds, clone=clone, dist=dist, div=div,
mid=mid, nums=nums, read=read, record=record, stats=stats)
 -- lib.lua: misc LUA functions
-- (c)2022 Tim Menzies <timm@ieee.org> BSD-2 licence
local l=()
l.b4=(); for k,v in pairs(_ENV) do l.b4[k]=v end
 -- Add 'x' to a list. Return 'x'.

function l.push(t,x) t[1+#t]=x; return x end
  -- Round
function l.rnd(n, nPlaces)
  local mult = 10^(nPlaces or 3)
  return math.floor(n * mult + 0.5) / mult end
 function 1.copy(t)
  if type(t) ~= "table" then return t end
  local u=(); for k,v in pairs(t) do u[k] = 1.copy(v) end
      - Return the 'p'-th thing from the sorted list 't'.
   function 1.per(t,p)
p=math.floor(((p or .5)*#t)+.5); return t[math.max(1,math.min(#t,p))] end
               o' generates a string from a nested table.
"o' generates a string from a nested table.
function | o(t) |
if type(t) -= "table" then return tostring(t) end |
local function show(k,v) |
if not tostring(k):find"o" then |
v = l.o(v) |
return #t=0 and string.format(".%s %s",k,v) or tostring(v) end end |
local u=(!) for k,v in pairs(t) do u[1+#u] = show(k,v) end |
if #t==0 then table.sort(u) end |
return (t._is or "")." | "end |
   -- 'oo' prints the string from 'o'.
function 1.oo(t) print(1.o(t)) return t end
   --- Convert string to something else.

function l.coerce(s)
local function coercel(s1)
if sl=="tme" then return true end
if sl=="tiske" then return false end
return sl end
return math.tointeger(s) or tonumber(s) or coercel(s:match*^%s*(-)%s*$*) end
   -- Iterator over csv files. Call 'fun' for each record in 'fname'. function l.csv(fname, fun)
      local srcs = io.input(fname)
while true do
local s = io.read()
if not s then return io.close(src) else
```

```
local t={} for s1 in s:gmatch("([^,]+)") do t[1+#t] = 1.coerce(s1) end
            fun(t) end end end
local t=()
s:gsub("un[-][%S]+[%s]+[-][-]([%S]+)[^\n]+=([%S]+)",
function(k,x) t[k]=1.coerce(x)end)
    t._help = s
return t end
  -- Update 't' from values after command-line flags. Booleans need no values
 -- (we just flip the defeaults).
function l.cli(t)
for slot,v in pairs(t) do
         v = tostring(v)
    v = tostring(v)
for n,x in ipairs(arg) do
   if x=="-". (slot:sub(l,1)) or x=="--"..slot then
   v = v=="fake" and "fune" or v=="fune" and "fake" or arg[n+1] end end
t[slot] = l.coerce(v) end
if t.help then os.exit(print("\n"..t._help.."\n")) end
 -- That's all folks. return 1
   \langle \overline{-} \rangle
 local l=require"lib"
local _=require"sam"
 local cli,o,oo,per,push,rnd = 1.cli,l.o,l.oo,l.per,l.push,l.rnd
 local add, adds, dist, div = _.add, _.adds, _.dist,_.div local mid, read, the = _.mid,_.read,_.the local Num, Sym = _.Num, _.Sym
local eg,fails = {},0
local function run(k, b4,out)
math.randomseed(the.seed)
b4 = 1.copy(the); out=eg[k](); the = 1.copy(b4);
print("!!!!!", k, out and "PASS" or "FAIL")
return out==true end
local function _egs( t)
  t={}; for k,_ in pairs(eg) do t[1+#t]=k end; table.sort(t); return t end
   unction eg.is()
print("\nExamples(\lua eg0.l\ua-fX)\\\nX=")
for _,k in pairs(_egs()) do print(\string.format(\" %-7s\",k)) end
return true end
 function eq.all()
    for _k in pairs(_egs()) do
   if k = "al" then fails = fails + (run(k) and 0 or 1) end end
return true end
 function eg.the() oo(the); return true end
function eg.ent( sym,ent)
  sym= adds(Sym(), {"a","a","a","a","b","b","c"})
  ent= div(sym)
    print(ent,mid(sym))
return 1.37 <= ent and ent <=1.38 end</pre>
 function eg.num( num)
   uneTune)
inm=Num()
for i=1,100 do add(num,i) end
local med,ent = mid(num), rnd(div(num),2)
print(mid(num),rnd(div(num),2))
return 50c med and med<= 52 and 30.5 <ent and ent <32 end
 function eg.bignum( num)
    num=Num()
the.nums = 32
    for i=1,1000 do add(num,i) end

oo(_.nums(num))

return 32==#num._has end
 function eg.read()
  oo(read("../../data/auto93.csv").cols.y); return true end
 function eq.dist( data)
    data=read(".J./data/auto93.csv")
for i=2, #data.rows do
    print(dist(data,data.rows[1], data.rows[i])) end
    return true end
the = cli(the)
if eg(the.eg) then run(the.eg) end
for k,v in pairs(_ENV) do if not l.b4[k] then print("?",k,type(v)) end end
os.exit(fails)
```

08/23/22 Page 3/3