```lua
#!/usr/bin/env lua
--                                    __
--                                   /\ \
--  _____          __          __    \ \ \/'\
-- /\ '__`\      /'__`\      /'__`\   \ \ , <
-- \ \ \L\ \    /\  __/     /\  __/    \ \ \\`\
--  \ \ ,__/    \ \____\    \ \____\    \ \_\ \_\
--   \ \ \/      \/____/     \/____/     \/_/\/_/
--    \ \_\
--     \/_/

local your, our={}, {b4={}, help=[[
peek.lua [OPTIONS]
(c)2022 Tim Menzies, MIT license (2 clause)
Understand N items after log(N) probes, or less.

  -file   ../../data/auto93.csv
  -best  .5
  -help  false
  -dull  .35
  -rest  3
  -seed  10019
  -rnd   %.2f
  -task  -
  -p     2]]}

for k,_ in pairs(_ENV) do our.b4[k] = k end
local any,asserts,cells,copy,fmt,go,id,main,many, map,new,o,push
local rand,randi,rnd,rogues,rows,same,settings,slots,sort,thing,things
local COLS,EG,EGS,NUM,RANGE,SYM
local class= function(t,  new)
  t.__index=t
  return setmetatable(t,{__call=function(_,...) return t.new(...) end}) end

-- Copyright (c) 2022, Tim Menzies
--
-- Redistribution and use in source and binary forms, with or without
-- modification, are permitted provided that the following conditions are met.
-- (1) Redistributions of source code must retain the above copyright notice,
-- this list of conditions and the following disclaimer.  (2) Redistributions
-- in binary form must reproduce the above copyright notice, this list of
-- conditions and the following disclaimer in the documentation and/or other
-- materials provided with the distribution.
--
-- THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS
-- IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
-- THE IMPLIED WARRANTIES OF MERCHNTABILITY AND FITNESS FOR A PARTICULAR
-- PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
-- CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
-- EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
-- PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
-- PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
-- LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
-- NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
-- SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE

-- ----------------------------------------------------------------------------
--
--           _
--       ___| | __ _ ___ ___  ___  ___
--      / __| |/ _` / __/ __|/ _ \/ __|
--     | (__| | (_| \__ \__ \  __/\__ \
--      \___|_|\__,_|___/___/\___||___/

AND=class{}
function AND.new() return new({cols={}, b=0,r=0,rows=nil}, AND) end

function AND.add(i,range,    at)
  i.rows = nil
  i.cols[range.col.at] = i.cols[range.col.at] or OR()
  i.cols[range.col.at]:add(range) end

function AND.all(i,goal,      both)
  function both(a,b,    c)
    c={};for id,row in pairs(a) do if b[id] then c[id]=row end end; return c end
  if not i.rows then
    for _,ors in pairs(i.cols) do
      i.rows = i.rows and both(i.rows,ors.rows) or ors.rows
      if #i.rows == 0 then return {} end end
    i:br(goal) end
  return i.rows end

function AND.br(i,goal,  rows)
  i.b, i.r = 0, 0
  for _,row in pairs(i:all()) do
    if row.class==goal then i.b=i.b+1 else i.r=i.r+1 end end end
-- ----------------------------------------------------------------------------
COLS=class{}
function COLS.new(t,     i,where,now)
  i = new({all={}, x={}, y={}},COLS)
  for at,s in pairs(t) do
    now = push(i.all, (s:find"^[A-Z]" and NUM or SYM)(at,s))
    if not s:find":" then
      push((s:find"-" or s:find"+") and i.y or i.x, now) end end
  return i end

function COLS.__tostring(i, txt)
  function txt(c) return c.txt end
  return fmt("COLS{:all %s\n\t:x %s\n\t:y %s", o(i.all,txt), o(i.x,txt), o(i.y,txt)) end

function COLS.add(i,t,      x)
  return map(i.all, function(col) x=t[col.at]; col:add(x);return x end) end

-- ----------------------------------------------------------------------------
EG=class{}
function EG.new(t) return new({has=t, id=id()},EG) end

function EG.__tostring(i) return fmt("EG%s%s %s", i.id,o(i.has),#i.has) end

function EG.better(i,j,cols)
  local s1,s2,e,n,a,b = 0,0,10,#cols
  for _,col in pairs(cols) do
    a  = col:norm(i.has[col.at])
    b  = col:norm(j.has[col.at])
    s1 = s1 - e^(col.w * (a-b)/n)
    s2 = s2 - e^(col.w * (b-a)/n) end
  return s1/n < s2/n end
-- ----------------------------------------------------------------------------
EGS=class{}
function EGS.new() return new({rows={}, cols=nil}, EGS) end

function EGS.__tostring(i) return fmt("EGS{#rows %s:cols %s", #i.rows,i.cols) end

function EGS.add(i,row)
  row = row.has and row.has or row
  if i.cols then push(i.rows,EG(i.cols:add(row))) else i.cols=COLS(row) end end

function EGS.bestRest(i)
  local best,rest,tmp,bests,restsFraction = {},{},{}
  i.rows = sort(i.rows, function(a,b) return a:better(b,i.cols.y) end)
  bests  = (#i.rows)^your.best
  restsFraction = (bests * your.rest)/(#i.rows - bests)
  for j,x in pairs(i.rows) do
     if     j      <= bests         then push(best,x)
```

```
    elseif rand() <  restsFraction then push(rest,x) end end
  return best,rest end

function EGS.clone(i,inits,    j)
  j = EGS()
  j:add(map(i.cols.all, function(col) return col.txt end))
  for _,x in pairs(inits or {}) do  j:add(x) end
  return j end

function EGS.file(i,f) for row in rows(f) do i:add(row) end; return i end

function EGS.mid(i,cols)
  return map(cols or i.cols.all, function(col)  return col:mid() end) end
-- ------------------------------------------------------------------------
NUM=class{}
function NUM.new(at,s, big)
  big = math.huge
  return new({lo=big, hi=-big, at=at or 0, txt=s or "",
              n=0, mu=0, m2=0, sd=0,
              w=(s or ""):find"-" and -1 or 1},NUM) end

function NUM.__tostring(i)
  return fmt("NUM{:at %s :txt %s :lo %s :hi %s :mu %s :sd %s}",
               i.at, i.txt,  i.lo, i.hi, rnd(i.mu), rnd(i:div())) end

function NUM.add(i,x,     d)
  if x~="?" then
    i.n = i.n+1
    d  = x - i.mu
    i.mu = i.mu + d/i.n
    i.m2 = i.m2 + d*(x-i.mu)
    i.lo = math.min(x,i.lo); i.hi = math.max(x,i.hi) end
  return x end

function NUM.div(i) return i.n <2 and 0 or (i.m2/(i.n-1))^0.5 end

function NUM.mid(i) return i.mu end

function NUM.norm(i,x) return i.hi-i.lo < 1E-9 and 0 or (x-i.lo)/(i.hi-i.lo) end

function NUM.ranges(i,j, bests,rests)
  local ranges,x,lo,hi,gap,tmp = {}
  hi  = math.max(i.hi, j.hi)
  lo  = math.min(i.lo, j.lo)
  gap = (hi - lo)/your.bins
  tmp = lo
  for j=lo,hi,goal do push(ranges,RANGE(i, tmp, tmp+gap)); tmp= tmp+gap end
  ranges[1].lo       = -math.huge
  ranges[#ranges].hi = math.huge
  for _,pair in pairs{{bests,"bests"}, {rests,"rests"}} do
    for _,row in pairs(pair[1]) do
      x = row.has[i.at]
      if x~= "?"  then
        ranges[(x - lo)//gap].stats:add(pair[2]) end end end end
-- ------------------------------------------------------------------------
OR=class{}
function OR.new() return new({ranges={}, rows={}}, OR) end

function OR.add(i,range)
  push(i.ranges,range)
  for id,row in pairs(range.rows) do i.rows[id] = row end end
-- ------------------------------------------------------------------------
RANGE=class{}
function RANGE.new(col,lo,hi,stats)
  return new({id=id(),col=col, lo=lo, hi=hi or lo,
              ys=stats or SYM(),rows={}},RANGE) end

function RANGE.__tostring(i)
  return fmt("RANGE{:col %s :lo %s :hi %s :ys %s}",
             i.col,i.lo,i.hi,o(i.ys)) end

function RANGE.add(i,x,y,row)
  assert(i.lo <= x and x < i.hi, "in range")
  i.ys[y]       = 1 + (i.ys[y] or 0)
  i.rows[row.id] = row end
-- ------------------------------------------------------------------------
SYM=class{}

function SYM.new(at,s)
  return new({at=at or 0,txt=s or "",has={},n=0,most=0,mode=nil},SYM) end

function SYM.__tostring(i)
  return fmt("SYM{:at %s :txt %s :mode %s :has %s}",
             i.at, i.txt, i.mode, o(i.has)) end

function SYM.add(i,x, inc)
  if x ~= "?" then
    inc = inc or 1
    i.n = i.n+inc
    i.has[x] = inc  + (i.has[x] or 0)
    if i.has[x] > i.most then i.most, i.mode = i.has[x], x end end
  return x end

function SYM.div(i)
  e=0;for _,v in pairs(i.has) do e=e - v/i.n*math.log(v/i.n,2) end; return e end

function SYM.mid(i) return i.mode end

function SYM.ranges(i,j,bests,rests)
  local tmp,out,x = {},{}
  for _,pair in pairs{{bests,"bests"}, {rests,"rests"}} do
    for _,row in pairs(pair[1]) do
      x = row.has[i.at]
      if x~= "?"  then
         tmp[x] = tmp[x] or SYM()
         tmp[x]:add(pair[2]) end end end
  for x,stats in pairs(tmp) do push(out, RANGE(i,x,x,stats)) end
  return out end
```

```lua
-- ---------------------------------------------------------------------------
--   __                  _   _
--  / _|_   _ _ __   ___| |_(_) ___  _ __  ___
-- | |_| | | | '_ \ / __| __| |/ _ \| '_ \/ __|
-- |  _| |_| | | | | (__| |_| | (_) | | | \__ \
-- |_|  \__,_|_| |_|\___|\__|_|\___/|_| |_|___/

fmt  = string.format
new  = setmetatable
same = function(x,...) return x end

function asserts(test,msg)
  msg=msg or ""
  if test then return print("PASS: "..msg) end
  our.fails = our.fails+1
  print("FAIL: "..msg)
  if your.Debug then assert(test,msg) end end

function copy(t,    u)
  if type(t)~="table" then return t end
  u={};for k,v in pairs(t) do u[k]=copy(v) end;return new(u,getmetatable(t)) end

function id() our.id = 1+(our.id or 0); return our.id end

function many(t,n, u) u={};for j=1,n do push(u,any(t)) end; return u end

function map(t,f,  u)
  u={};for _,v in pairs(t) do u[1+#u]=(f or same)(v) end; return u end

function o(t,f,   u,key)
  key= function(k)
         if t[k] then return fmt(":%s %s", k, rnd((f or same)(t[k]))) end end
  u = #t>0 and map(map(t,f),rnd) or map(slots(t),key)
  return "{"..table.concat(u, " ").."}" end

function rand(lo,hi)
  your.seed = (16807 * your.seed) % 2147483647
  return (lo or 0) + ((hi or 1) - (lo or 0)) * your.seed / 2147483647 end

function randi(lo,hi) return math.floor(0.5 + rand(lo,hi)) end

function push(t,x)  table.insert(t,x); return x end

function rnd(x)
  return fmt(type(x)=="number" and x~=x//1 and your.rnd or"%s",x) end

function rows(file,      x)
  file = io.input(file)
  return function()
    x=io.read(); if x then return things(x) else io.close(file) end end end

function main(      defaults,tasks)
  tasks = your.task=="all" and slots(go) or {your.task}
  defaults=copy(your)
  our.failures=0
  for _,x in pairs(tasks) do
    if type(our.go[x]) == "function" then our.go[x]() else print("?", x) end
    your = copy(defaults) end
  rogues()
  return our.failures end

function rogues()
  for k,v in pairs(_ENV) do
    if not our.b4[k] then print("?",k,type(v)) end end end

function settings(help,   t)
  t={}
  help:gsub("\n [-]([^%s]+)[^\n]*%s([^%s]+)", function(slot, x)
    for n,flag in ipairs(arg) do
      if   flag:sub(1,1)=="-" and slot:match("^"..flag:sub(2)..".*")
      then x=x=="false" and "true" or x=="true" and "false" or arg[n+1] end end
    t[slot] = thing(x) end)
  if t.help then print(t.help) end
  return t end

function slots(t,u) u={};for x,_ in pairs(t) do u[1+#u]=x end;return sort(u) end
```

```lua
function sort(t,f)  table.sort(t,f); return t end

function thing(x)
  x = x:match"^%s*(.-)%s*$"
  if x=="true" then return true elseif x=="false" then return false end
  return tonumber(x) or x end

function things(x,sep,  t)
  t={};for y in x:gmatch(sep or"([^,]+)") do t[1+#t]=thing(y) end; return t end
-- ---------------------------------------------------------------------------
--  _            _
-- | |_ ___  ___| |_ ___
-- | __/ _ \/ __| __/ __|
-- | ||  __/\__ \ |_\__ \
--  \__\___||___/\__|___/

our.go, our.no = {},{}; go=our.go
function go.settings() print("your",o(your)) end

function go.sample() print(EGS():file(your.file)) end

function go.clone( a,b)
  a= EGS():file(your.file)
  b= a:clone(a.rows)
  asserts(o(a.cols.all[1])==o(b.cols.all[1]),"cloning")
end

function go.sort(   i,a,b)
  i   = EGS():file(your.file)
  a,b = i:bestRest()
  a,b = i:clone(a), i:clone(b)
  print("all",  o(i:mid(i.cols.y)))
  print("best", o(a:mid(a.cols.y)))
  print("rest", o(b:mid(b.cols.y)))
end

your = settings(our.help)
os.exit( main() )
```