```lua
local R = require
local _,the,COLS,BIN               = R"lib", R"the", R"COLS", R"BIN"
local map,sort,upl,items,push,norm = _.map,_.sort,_.upl,_.items,_.push,_.norm
local items,slice,o,oo,sort,many   = _.items,_.slice,_.o,_.oo,_.sort,_.many
local class,OBJ                    = _.class, _.OBJ

local EGS = class("EGS",OBJ)
function EGS:new() self.rows, self.cols = {}, nil end

function EGS:adds(y) for x in items(y) do self:add(x) end; return self end

function EGS:add(row)
  if not self.cols then self.cols = COLS(row)
                   else push(self.rows, self.cols:add(row)) end end

function EGS:mid(cols)
   local mid=function(col) return #self.rows==0 and 0 or col:mid() end
   return map(cols or self.cols.y, mid) end

function EGS:div(cols)
   local div=function(col) return #self.rows==0 and 0 or col:div() end
   return map(cols or self.cols.y, div) end

function EGS:clone(rows)
  local out = EGS()
  out:add(self.cols.names)
  for _,row in pairs(rows or {}) do out:add(row) end
  return out end

function EGS:dist(row1,row2)
  local d, n = 0, 0
  for _,col in pairs(self.cols.x) do
    n = n + 1
    d = d + col:dist(row1[col.at], row2[col.at])^the.p end
  return (d/n) ^ (1/the.p) end

function EGS:better(row1,row2)
  local s1, s2, n, e = 0, 0, #self.cols.y, math.exp(1)
  for _,col in pairs(self.cols.y) do
    local a = norm(col.lo, col.hi, row1[col.at] )
    local b = norm(col.lo, col.hi, row2[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

function EGS:bins(other)
  local out = {}
  for n,left in pairs(self.cols.x) do
    local right = other.cols.x[n]
    local tmp   = left:bins(right,BIN)
    if #tmp > 1 then for _,bin in pairs(tmp) do push(out,bin) end end end
  return out end

function EGS:bestRest()
  self.rows = sort(self.rows, function(a,b) return self:better(a,b) end)
  local n = (#self.rows)^the.best
  return slice(self.rows, 1,           n),      -- top n things
               many( self.rows, n*the.rest, n+1) end -- some sample of the rest

-- function egs.xplain(i)
--   best, rest = egs.bestRest(i)
--   return egs.contrasts(i, best,rest) end

return EGS
```