```lua
local _,ako,COL = require"lib", require"ako", require"COL"
local map,slots,class,ent = _.map,_.slots,_.class,  _.ent

local SYM = class("SYM",COL)
function SYM:new(at,name)
  self:super(at,name)
  self.has, self.most, self.mode = {}, 0, nil end

function SYM:add1(x,inc)
  self.has[x] = inc + (self.has[x] or 0)
  if self.has[x] > self.most then
    self.mode, self.most = x, self.has[x] end end

function SYM:mid()      return self.mode end
function SYM:div()      return ent(self.has, self.n) end
function SYM:same(x,y)  return x==y end
function SYM:dist1(x,y) return self:same(x,y)  and 0 or 1 end

function SYM:like1(x,prior)
  return ((i.has[x] or 0) + the.M*prior)/(self.n + the.M) end

function SYM:merge(other,      out)
  out = SYM(self.at, self.name)
  for x,n in pairs(self.has)  do out:add(x,n) end
  for x,n in pairs(other.has) do out:add(x,n) end
  return out end

function SYM:bins(other, BIN)
  local out = {}
  local function known(x) out[x] = out[x] or BIN(self.at, self.name, x,x) end
  for x,n in pairs(self.has)  do known(x); out[x].ys:add("left", n) end
  for x,n in pairs(other.has) do known(x); out[x].ys:add("right", n) end
  return map(slots(out), function(k) return out[k] end) end

return SYM
```