```
(defpackage :tiny (:use :cl))
(in-package :tiny)
(load "ljh")
(defvar *opt*
  (settings  "TOYIN: do stuff
      (c) 2022 Tim Menzies, BSD-2 clause license "
  '((file  "-f"  "help file          " "../../data/auto93.lisp")
    (help  "-h"  "show help          " nil)
    (keep  "-K"  "items to keep      " 256)
    (k     "-k"  "nb low attributes classes" 1)
    (m     "-m"  "nb low frequency classes " 2)
    (seed  "-s"  "random number seed   " 10019)
    (go    "-g"  "start up action     " ls))))

(defmacro ! (key) `(cdr (assoc ',key *opt*)))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(defstruct+ sym  (txt "") (at 0) kept)
(defstruct+ num  (txt "") (at 0) kept ok (w 1))
(defstruct+ cols names all x y klass)
(defstruct+ data rows about)
(defstruct+ row  cells _about)

(defun make-sym (s n) (%make-sym :txt s :at n))
(defun make-num (s n) (%make-num :txt s :at n :w (if (equal #\- (charn s)) -1 1)))
(defun make-row (about l)   (%make-row :cells l :_about about))

(defun make-cols (lst)
  (let (all x y kl (pos -1))
    (dolist (s lst (%make-cols :names lst :all (reverse all) :x x :y y :klass kl))
      (let* ((what (if (eq #\$ (char0 s)) 'make-num 'make-sym))
             (col  (funcall what s (incf pos))))
        (push col all)
        (unless (eq #\~ (charn s))
          (if (member (charn s) '(#\! #\- #\+)) (push  col y) (push  col x))
          (if (eq #\! (charn s)) (setf kl col)))))))

(defun make-data (names &optional src (i (%make-data :about (make-cols names))))
  (if (stringp src)
      (with-lines src (lambda (line) (add i (cells line))))
      (dolist (row src) (add i row)))
 i)

(print (make-row 12 '(1 2 3 4)))
(print (make-data '($aa bb!~ cc+)))
(print *opt*)
; ; (defmethod clone ((d data) &optional src) (make-data (? d about names) src))
; ;(reads "../../data/auto93.lisp" 'print)
```