```python
     from __future__ import division,print_function
     import sys,random,re
     sys.dont_write_bytecode =True

5    def cached(f=None,cache={}):
       if f:
         def wrapper(**d):
           tmp = cache[f.__name__] = f(**d)
           return tmp
10       return wrapper
       else:
         for x in cache: print(x,cache[x])

     #################################################
15   @cached
     def genic0(**d):
       def tiny(u,w):
         return u <  w.opt.era/w.opt.k
       return o(
20       k=10,
         era=1000,
         tiny= tiny,
         num='$',
         klass='=',
25       seed=1).update(**d)

     @cached
     def rows0(**d): return o(
       skip="?",
30     sep  = ',',
       bad = r'(["\'\t\r\n]|#.*)'
       ).update(**d)

     #################################################
35   rand= random.random
     seed= random.seed

     def say(c):
       sys.stdout.write(str(c))

40   def fun(x):
       return x.__class__.__name__ ≡ 'function'

     def g(lst,n=3):
45     for col,val in enumerate(lst):
         if isinstance(val,float):
           val = round(val,n)
         lst[col] = val
       return lst

50   def printm(matrix):
       s = [[str(e) for e in row] for row in matrix]
       lens = [max(map(len, col)) for col in zip(*s)]
       fmt = '\t'.join('{{:{}}}'.format(x) for x in lens)
55     for row in [fmt.format(*row) for row in s]:
         print(row)


     class o:
60     "Define a bag of names slots with no methods."
       def __init__(i,**d): i.update(**d)
       def update(i,**d):
         i.__dict__.update(**d); return i
       def __repr__(i)   :
65       def name(x):
           return x.__name__ if fun(x) else x
         d   = i.__dict__
         show = [':%s=%s' % (k,name(d[k]))
                   for k in sorted(d.keys() )
                   if k[0] is ¬ "_"]
70       return '{'+' '.join(show)+'}'
```

```python
75   #################################################
     def rows(file,w=None):
       """Leaps over any columns marked 'skip'.
       Turn strings to numbers or strings.
       Kill comments. Join lines that end in 'sep'."""
80     w = w ∨ rows0()
       def atom(x):
         try : return int(x)
         except ValueError:
           try : return float(x)
85         except ValueError : return x
       def lines():
         n,kept = 0,""
         for line in open(file):
           now  = re.sub(w.bad,"",line)
90         kept += now
           if kept:
             if ¬ now[-1] ≡ w.sep:
               yield n, map(atom, kept.split(w.sep))
               n += 1
95             kept = ""
       todo = None
       for n,line in lines():
         todo = todo ∨ [col for col,name
                           in enumerate(line)
100                        if ¬ w.skip in name]
         yield n, [ line[col] for col in todo ]

     def header(w,row):
       def numOrSym(val):
105       return w.num if w.opt.num in val else w.sym
       def indepOrDep(val):
         return w.dep if w.opt.klass in val else w.indep
       for col,val in enumerate(row):
         numOrSym(val).append(col)
110       indepOrDep(val).append(col)
         w.name[col] = val
         w.index[val] = col

     def data(w,row):
115     for col in w.num:
         val = row[col]
         w.min[col] = min(val, w.min.get(col,val))
         w.max[col] = max(val, w.max.get(col,val))

120   def indep(w,cols):
       for col in cols:
         if col in w.indep: yield col

     #################################################
125   def nearest(w,row):
       def norm(val,col):
         lo, hi = w.min[col], w.max[col]
         return (val – lo ) / (hi – lo + 0.00001)
       def dist(centroid):
130       n,d = 0,0
         for col in indep(w, w.num):
           x1,x2 = row[col], centroid[col]
           n1,n2 = norm(x1,col), norm(x2,col)
           d    += (n1 – n2)**2
135         n    += 1
         for col in indep(w, w.sym):
           x1,x2 = row[col],centroid[col]
           d    += (0 if x1 ≡ x2 else 1)
           n    += 1
140       return d**0.5 / n**0.5
       lo, out = 10**32, None
       for n,(_,_,_,centroid) in enumerate(w.centroids):
         d = dist(centroid)
         if d < lo:
145         lo,out = d,n
       return out
```

```python
     def move(w,new,n):
150     u0,u,age,old = w.centroids[n]
       u1 = 1
       out = [None]*len(old)
       for col in w.sym:
         x0,x1 = old[col], new[col]
155       out[col] = x1 if rand() < 1/(u0+u1) else x0
       for col in w.num:
         x0,x1= old[col], new[col]
         out[col] = (u0*x0 + u1*x1)/ (u0+u1)
       w.centroids[n] = (u0 + u1,u+u1, age+1, out)

160   def less(w) :
       b4 = len(w.centroids)
       w.centroids = [(1,u,dob,row)
                      for u0,u,dob,row in
165                      w.centroids
                      if ¬ w.opt.tiny(u0,w)]
       print("–",b4 – len(w.centroids))

     def genic(src='data/diabetes.csv',opt=None):
170     w = o(num=[], sym=[], dep=[], indep=[],
           centroids=[],
           min={}, max={}, name={},index={},
           opt=opt ∨ genic0())
       for n,row in rows(src):
175       if n ≡ 0:
           header(w,row)
         else:
           data(w,row)
           if len(w.centroids) < w.opt.k:
180           say("+")
             w.centroids += [(1,1,n,row)]
           else:
             move(w,row,nearest(w,row))
             if 0 ≡ (n % w.opt.era):
185             say(n)
               less(w)
       return w,sorted(w.centroids,reverse=True)

     def report(w,clusters):
190     print("")
       cols = w.index.keys()
       header = sorted(w.name.keys())
       header= [w.name[i] for i in header]
       matrix = [['id','lastGen','allGen','dob'] + header]
195     for m,(u0,u,age,centroid) in enumerate(clusters):
         if ¬ w.opt.tiny(u0,w):
           matrix += [[m+1,u0,u,age] + g(centroid,2)]
       printm(matrix)

200   if __name__ ≡ '__main__':
       src='data/diabetes.csv'
       if len(sys.argv) ≡ 2:
         src= sys.argv[1]
       opt=genic0(era=100,k=16,seed=200)
205     seed(opt.seed)
       report(*genic(src,opt))


     cached()
```