**Kmean baseline input**

Function recalcularCentre (Parameters: pointer to punts, N, D, C, pointer to

Centroides, pointer to PC, pointer to Sep)

{

Set i, j, k, x to zero

Set pointer v to malloc (size of D)

Set r to zero

Loop (Set i to zero; Check i less than to D; Increment i with one)

{

Loop (set x to zero; Check x less than to D; Increment x with one)

{

Set array v [index x] to zero

}

Loop (set j to zero; check j less than array Sep [index i]; Increment j with one)

{

Loop (set k to zero; check k less than to D; Increment k with one)

{

Set array v [index x] with sum of v [index x] and (Type casting

double) array Punts [ array PC [size is sum of

pointer to i, N, j] set sum of pointer D and k]

}

Loop (set x to zero; check x less than to D; Increment x with one)

{ Set array v [index x] with division of array v [index x] and

(type casting with double) array Sep [index i]

If (array v [index x] not equal to Centroides)

Set r to 1

Set Centroides to array v [index x]

}

}

Free array v

Return (r)

}

Function read\_CSV\_data (Parameters: pointer filename, double pointer row\_name,

Pointer Punts, N, D, C, pointer Centroides)

{

Set pointer fp to filename

If (not fp)

{

Print (filename)

Return (one)

}

Print (filename)

Set i, j, n\_read, n\_total\_idx to zero

Set array line [size one thousand and twenty-four]

Call function (Arguments: lines, size of(line), fp)

While (Call function (Arguments: lines, size of(line), fp) )

{

Set token to line

Set j, n\_each\_in\_line to zero

While (token not equal to null)

{

If (strcmp (token) is equal to zero)

Stop

If (j is equal to zero)

{

Call function strcpy (Arguments: row\_name [index i], token)

Set j to sum of j and one

Set token to null

Continue

}

Set array Punts [index n\_total\_idx plus one] to token

Set n\_each\_in\_line to sum of n\_each\_in\_line and one

Set j to sum of j and one

If (n\_each\_in\_line is greater than equal to D)

Stop

Set token to null

}

If (n\_each\_in\_line not equal to D)

{

Print (n\_each\_in\_line)

Print (D)

Return (two)

}

Set i to sum of i and one

Set n\_read to sum of n\_read and one

If (n\_read greater than N)

Stop

Call function fclose (Arguments: fb)

Call function assert (n\_read is equal to N)

Set N to n\_read

Print (N, D, filename)

Loop (set i to zero; check i less than C; Increment i with one)

{

Set k to remainder or rand and N

Loop (set j to zero; check j less than D; Increment j with one)

{

Set Centroides to Punt

}

}

Loop (set i to zero; check i less than C; Increment i with one)

{

Loop (set j to zero; check j less than D; Increment j with one

Print (Centroides)

}

Return (zero)

}

Function mmin (parameters: pointer to v, C)

{

Set m to array v [index zero]

Set i to one and r to zero

Loop (set i to one; check i less than to C; Increment i with one)

{

If (m greater than array v [index i]

{

Set r to i and Set m to array v [index i]

}

}

Return (r)

}

Function PointsToCentroides (Parameter: pointer to Punts, N, D, C, pointer to

Centroides, pointer to PC, pointer to Sep)

{

Set i, j, k to zero

Set pointer dist to malloc (size of C)

Loop (set i to zero; check i less than to C; Increment i with one)

{

Set array Sep [index i] to zero

}

Loop (set i to zero; check i less than to N; Increment i with one)

{

Loop (set j to zero; check j less than to C; Increment j with one)

{

Set dist [index j] to zero

Loop (set k to zero; check k less than to D; Increment k with one)

{

Set array dist [index j] to (difference of Punts and Centroides for i)

multiplication of (Difference of punts and Centroides for j)

}

}

Set m to Call function mmin (argument: dist, C)

Set array PC to i

Set array Sep to sum of array Sep and one

}

Free dist

}

Start

Function main (Parameters: argc, pointer to argv)

{

Declare N, D, C

Set pointer file to malloc [size one thousand and twenty-four]

If (value of argc greater than five)

{

Call function (Argument: file)

Set C to four

Set N to one million

Set D to sixty-eight

Print (argv [index zero])

Print (C, N, D)

}

Else {

Call function (Argument: file, argv [index one])

Set C to call function atoi (Argument: argv [index two]

Set N to call function atoi (Argument: argv [index three])

Set D to call function atoi (Argument: argv [index four])

}

Set pointer Punts to malloc (size of pointer N and D)

Set pointer Centroides to malloc (size of pointer C and D)

Set pointer PC to malloc (size of pointer N and C)

Set pointer Sep to malloc (size of pointer C)

Set cont and final to zero

Set k to zero

Set double pointer row\_name to malloc (size of N)

Loop (set k to zero; check k less than N; Increment k with one)

Set array row\_name [index k] to malloc (size of ten)

Print (N, D, C)

If (call function read\_CSV\_data (Arguments: file, row\_name, Punts, N, D,

C, Centroides) is greater than to zero)

Exit(one)

Do

{

Call function PointsToCentroides (Arguments: Punts, N, D, C, Centroides, Sep)

Set cont to sum of cont and one

Set final to call function recalcularCentre (Arguments: Punts, N, D, C,

Centroides, PC, Sep)

}

While (final or cont less than to two hundred)

{

Print (cont)

Print (final)

Exit (zero)

Set i and j to zero

Loop (set i to zero; check i less than C; Increment i with one)

{

Loop (set j to zero; check j less than D; Increment j with one)

{

Print (Centroides)

}

}

}

Set array line [size one thousand and twenty-four]

Set pointer fp to file

Call function fgets (Arguments: line, size of (line), fp)

Call function fclose (Arguments: fp)

Set pointer result\_file to result\_csv

If (not result\_file)

Return (one)

fprint (result\_file, C)

Loop (set k to zero; check k less than to C; Increment k with one)

fprint (result\_file, k plus one, Sep [index k])

Loop (set i to zero; check i less than to C; Increment i with one)

{

fprint (result\_file, i plus one)

fprint (result\_file, line)

Loop (set j to zero; check i less than to Sep [index i]; Increment j with one)

{

fprint (result\_file, row\_name)

Loop (set k to zero; check k less than to D; Increment k with one)

{

fprint (result\_file, Punts)

}

}

}

}

Call function fclose (Arguments: result\_file)

Call function free (Argument: file);

Call function free (Argument: Punts);

Call function free (Argument: Centroides);

Call function free (Argument: PC);

Call function free (Argument: Sep);

Call function free (Argument: row\_name);

Exit (zero)