

1. Why MStrat?

MSTRAT is a program for building germplasm core collections by maximizing allelic or phenotypic richness against given individuals number.

Schoen and Brown (1993) introduced this methodology of constitution of core collection.

As implemented initially, the so-called M (or maximization) strategy samples a core collection by maximizing the number of observed alleles at the marker loci. We have recently extended the M strategy to qualitative and quantitative variables.

The program may be downloaded and used free of charge. The program may be used for non-commercial purposes. When using this program for published work, the following reference should be cited:

Gouesnard B., Bataillon T.M., Decoux G., Rozale C., Schoen D.J., David J.L (2001). "MSTRAT: An Algorithm for Building Germ Plasm Core Collections by Maximizing Allelic or Phenotypic Richness". Journal of Heredity : 92(1) : 93-94

2. How to install MStrat

First of all, if you want to use MStrat in interface mode (see: "Running MStrat" section) you will have to download TCL environment at <http://www.tcl.tk> and install it before launching MStrat.

For Windows: Download the MStrat installation program, launch-it and follow the instructions

For Unix: Download the MStrat archive and unzip-it wherever you want, then you should compile "ether.c" using -lm option, for example using GNU gcc compiler : `gcc -o ether ether.c -lm`

Very important: In some cases MStrat could reject working with directory containing spaces characters. Then, if you get a "file not found" error beware that your files are stored in a directory such as "c:/MyPrograms/MStrat/MyFiles" instead of "c:/Program files/MStrat/My files"

3. Input files

To run, MStrat needs some information given in few input files such as:

a. Data file

The data file contains the description of accessions for each variable and allows several individuals per accession. The separator is the blank. So, avoid any space within alphanumeric data.

First column : accession identifier (called “code”) in Numeric. You can give non adjacent and non ordered values.

Second column : number of individuals per accession in Numeric. Put the number of individual to 1 if you have only one individual by accession on the whole collection. MSTRAT will maximize the diversity of accessions, using the range covered by all their individuals...

Third column up to the end : variables in numeric or in alphanumeric format, as many as variables you want (<=500).

The two first columns are **COMPULSARY** in numeric. You can introduce other identification variables of the accessions (such as the name of accessions) in numeric or alphanumeric but these variables must be declared as inactive variables. The spacer between fields is the blank. So, don't put blank characters before the first column or inside a variable.

Mstrat handle missing data. The code of missing data is the point. Then in the program, the code of missing data is the number 9999. So, the value 9999 is to be avoided. It will be read as a missing data. If an accession have missing data for a variable, the accession brings no richness for this variable. Accessions having missing data will be retained in the core collection only they are original for the another variables.

Example (ficdon.dat)

[illegible]

b.Variables file

The variable file gives the heading of the columns of the data file. Different types of variables are defined according to their use in the MSTRAT program. Variables that are used in the optimization of the richness are called active variables. Target variables are not used in the sampling but are used to calculate the diversity score of the core collection. The variable file could be easily modified with the Tcl interface

When using a data file for the first time, just build a simple file declaring the name of each variable, one line by variable.

Don't put the two variables: code and individu

Example (ficvar.var)

NOM
REGION
DEP
PREC
T
C
GDUFE
M
CONI
NRG
P1000G
LOC1a1
LOC1a2
LOC1a5
LOC1a7
LOC3
LOC4

After clicking on the button “Variables”, a new window appears. On the screen, the name of the variables is given with 7 columns called : inactive, quali, quanti, active, target, weight, classes (ficvar).

Firstly, variables can be either inactive (inactive in Mstrat.tcl) or else (commented after). You have to declare as inactive any variable you've got in your file that is not interesting for building or validating the core. For example in ficdon.txt, the variables : NOM, REGION, DEP, T, C, NRG are inactive. For inactive variables, click on the first column (grey round) called “inactive”.

When the variables are not inactive they can be either qualitative ('qual' in Mstrat) or quantitative ('quant') . Qualitative data ('qual') are mainly markers with allelic code (1,..N), or any discrete variables (such as color...). The maximum number of classes is defined to 1000 in the C program. Quantitative data ('quant') are continuously varying data as usual morphologic traits. When they are quantitative you have to declare the number of classes (5 is value default) you want in the 'classes' box (last box). The interval between two adjacent classes is constant. It is given by the (maximum value - minimum value) / number of classes.

Weight ('weight') can be applied on active or target variables (in Numeric, real). By default, a weight of 1 is given for each variable.

Finally, you have to declare if you want variables to be active in the sampling of the core collection ('active' in the Mstrat.tcl window) or target ('target'). Active variables are those called Markers by Schoen and Brown. 'target' variable means that Mstrat will compute the score realized on these variables using active variables. This allows to validate the Mstrat approach.

After saving, the file (ficvar.txt) will record the following description when recalled by "variable" button. The output variable file have 2 lines more than the first variable file :

- First line : name of accession variable followed by 0
- Second line : name of individual variable followed by 0

Following lines :

- first column : name of variable
- second column : 1 (inactive), 2 (qualitative data), 3 (quantitative data)
- third column : 1 (active or marker), 0 (no active variable)
- forth column : 1 (target variable), 0 (no target variable)
- fifth column : (weight)
- sixth column : number of classes

You can modify the file using a simple text editor; but, don't add blanks in the file.

example :

```
code 0
individu 0
NOM 1 0 0 1 5
REGION 1 0 0 1 5
DEP 1 0 0 1 5
PREC 2 1 0 1 5
T 1 0 0 1 5
C 1 0 0 1 5
GDUFEM 3 1 0 1
12
CONI 3 1 0 1 12
NRG 1 0 0 1 12
P1000G 3 1 0 1 12
LOC1a1 2 0 1 1 5
LOC1a2 2 0 1 1 5
LOC1a5 2 0 1 1 5
LOC1a7 2 0 1 1 5
LOC3 2 0 1 1 5
LOC4 2 0 1 1 5
```

c. Kernel file

The kernel file gives a subset of compulsory accessions in the core collection. The sampling procedure optimizes the diversity criterion of sub sample individuals that complete at best the diversity of the kernel set. Kernel core accessions can be, for example, historical or reference accessions. The list of kernel core accessions can be built up with the Tcl interface.

The kernel file can be built from a file containing only one column given the total codes of accessions.

Under the TCL interface, indicate the presence of an accession in the kernel by clicking the button of the accession code.

This output file contains two columns:

the first one presents accessions codes and the second one gives the presence (0) or not (1) in the kernel core.

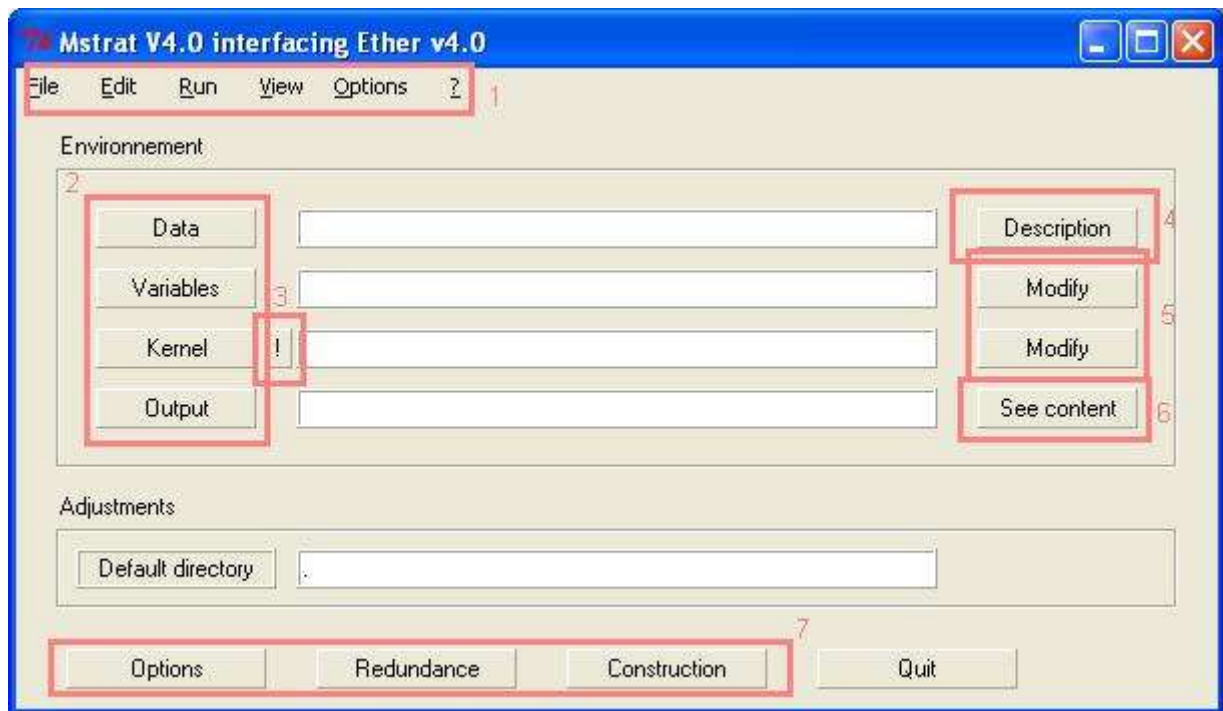
Example (ficker.ker)

3 0
5 0
6 0
10 0
11 0
12 0
14 0
15 1
16 0
...

4. Running MStrat

There are two ways of running MStrat:

a. Interface



- 1. Menu bar, you can access to all functions by using it
- 2. Click on these buttons to select files location
- 3. If you want to select an other kernel file, click on this "!" first, otherwise an error will occure
- 4. Click on this button to see the description of the data file (data file and variable file location must have been selected)
- 5. Click on these buttons to modify the variables file or the kernel file
- 6. Click on this button to see the result of a redundancy or a construction.
- 7. Click on one of these buttons to access different features of the program

Very important: Beware not launching a construction or a description after a redundancy with the same output filename, otherwise you will erase the redundancy result!

b.Console running

If you want to use MStrat without installing TCL or if you prefer using consoles

Thanks to ether.exe you can run MStrat in batch mode.

The “ param.tmp ” file containing needed information is considered as argument. This file is automatically built up by MStrat interface and placed in the input files directory.

The description command is : ether -d param.tmp

The construction batch commands are:

- ether -f param.tmp
- R\bin\Rterm.exe -no save -quiet -slave < vargen\in.txt
- ether -f param.tmp

The redundancy batch commands are :

- ether param.tmp
- R\bin\Rterm.exe -no save -quiet -slave < vargen\in.txt
- ether param.tmp

• Param.tmp

This file contains :

- 1st line : the name of the output file
- 2nd line : the name of the variable file
- 3th line : the name of the data file
- 4th line : seeds file
- 5th line : the core size
- 6th line : the number of replicates
- 7th line : number of iterations.
- 8th line : choice of second criteria of maximization (0 : generalized variance, 1: Nei indice, 2: Shannon indice)
- 9th line : flag for debug
- 10th line : flag for popdep
- 11th line : flag for axij
- 12th line : data directory
- 13th line : step value
- 14th line : start of the accurate range
- 15th line : end of the accurate range
- 16th line : the name of kernel file

c. Example of file "param.tmp"

1:	C:/Mstrat/data/Ficsor.out	<-----	output file
2:	C:/Mstrat/data/Ficvar.var	<-----	variable file
3:	C:/Mstrat/data/Ficdon.dat	<-----	data file
4:	C:/Mstrat/seeds.txt	<-----	seed file
5:	10	<-----	core collection size
6:	2	<-----	number of replicates
7:	15	<-----	number of iterations
8:	0	<-----	2nd criteria choice (0:vargen;1:Nei;2:Shannon)
9:	1	<-----	flag for debug
10:	0	<-----	flag for popdep
11:	0	<-----	flag for axij
12:	C:/Mstrat/data	<-----	data directory
13:	-1	<-----	Step value (-1 => auto)
14:	10	<-----	custom stepping start value (-1 => auto)
15:	20	<-----	custom stepping end value (-1 => auto)
16:	C:/Mstrat/ficker.ker	<-----	kernel file

5. Core collection constitution

Preliminary step: check of files

The input files could be rapidly controlled by building up a core collection of a small size. Errors are printing in a file (See details previously given). Use the description mode to control the presence of missing data and the description of variables.

1st step: study of the collection

The Tcl interface allows to view graphs of richness against the number of individuals in the core. Two plots are produced: random and M curves. The random curve indicates the extent of potential redundancy in the collection. A convex relationship indicates redundancy whereas a linear relationship indicates zero redundancy in the collection. The choice of variables defining richness and the number of classes in the variables influence the expected amount of redundancy.

The choice of active variables and the number of classes in variables could be done on two criterions: the shape of the curve of richness against the number of individuals in the core and the response on 'target' variables. With a nearly linear curve, the sample which optimises the richness will be difficult to find. With a very convex curve, the optimal sample will have a very small size and beyond this size, various samples will satisfy the optimum of richness. The choice of variables and number of classes could also be defined on the response on other variables. Higher will be the response on target variables; higher will be probably the response on other loci.

After the choice of variables and the number of classes in the variables, the inflection point of the M curve provides the optimal size for a core collection.

2nd step: Core collection constitution

Several samples are obtained according to the requested number of replicates. The final core collection could be chosen on the response on target variables.

To define a final core collection, you can ask a great number of replicates and retain in the final core collection the accessions which are sampled a great number of times.

6. Description

Press the button 'Description' at the end of the line given the data file to get description. The file "Descri.txt" gives the following information on data file and variable file :

- a. Information on number of accessions, number of variables (inactive, quantitative, qualitative, active and target)
- b. Information of data file : number of missing data, maximum score on active variables and on target variables, a list of accession which are unique for a class of variable.

7.Redundancy

To obtain several core collections, indicate the number of replicates

The Mstrat program samples different sizes of core collections. The step of the redundancy curve changes of value in regard to the subcore size

If you enter "-1" in the step menu, the program will choose automatically the range and step default values.

Given "n" as number of accessions of the data file, we have got by default:

- a. From the number of accession in the kernel to $n/4$, the step between two samples is $n/30$
- b. from $n/4$ to $n/2$, the step is $n/10$
- c. from $n/2$ to n , the sampling is only done for n

Since the 4.1 version we can change the step in a precise interval by selecting the start and the end of the interval so as the step after pressing the "Step" button

Iterations settings will influence the results of the maximization algorithm

“ Graph average ” button allows the visualization of means of richness over the replicates either for active variables or for target variables. Two curves are given : one for random sampling and the other for optimization by Mstrat.

“ Graph cloud ” button allows the visualization of all the points (all the replicates) The coordinates of the points of the curves are defined as integer (not as a real).

The outfile contains from line 4 to the end :

- a. 1st column : OPT or RAN for optimize or random
- b. 2nd column : the number of replicates
- c. 3th column : the size of the core collection
- d. 4th column: the score of active variables
- e. 5th column : the score on target variables

8.Construction of Core Collections

To indicate the core size, click on the cursor or press the bar called “ core size ”. The maximum of the core size can be changed. Choose the number of iteration by the same way. The number of iteration should be larger with the size of the core collection.

The output file contains from line 4 to the end :

- a. 1st column : number of replicates
- b. 2nd column: number of accessions
- c. 3th column : score on active variables
- d. 4th column : values of Nei or Shannon indice (when one of these criteria is chosen)
- e. 5th column : values on target variables
- f. 6th column : number of iterations

9.Options

By selecting the "set options" item of the Options menu in the top menu bar, as so as pressing the "options" button at the bottom of the main window, you can access different settings about:

a.Second criterion

When different samples have the same richness, a second criteria of optimization is used.

Three criteria are proposed :

- **Nei indice**

if p_{ij} represents the i class frequency of the j variable, the Nei diversity indice is:

$$I_{Nei,j} = 1 - \sum_i p_{ij}^2$$

The indice for all of variables will be sum of indices for each variable

$$I_{Nei} = \sum_j I_{Nei,j}^2$$

What's interesting with Nei and Shannon indices is that they are both based on classes frequency, which is immediatly get owing to AxijFreq matrix.

Moreover they permit not to favour the more frequent classes when drawing competing accessions.

Indeed, subcore collections are favoured when they contain fair-distributed classes, because these two indices are leveled-up with frequency similarity.

- **Shannon indice**

In the Nei indice way, the indice of all of variables is the sum of indices for each variable.

$$I_{Shannon,j} = - \sum (p_{ij} \ln p_{ij})$$

- **Generalized variance**

For the generalized variance criteria, a principal component analysis is performed with R software on complete data file.

Generalized variance is calculated from active quantitative and qualitative variables stored into Popdep matrix. The data file has to be complete in order to perform multivariable analysis. First of all, qualitative variables are processed by a correspondence factor analysis.

Statistical calculations are performed with the R software.

The process is :

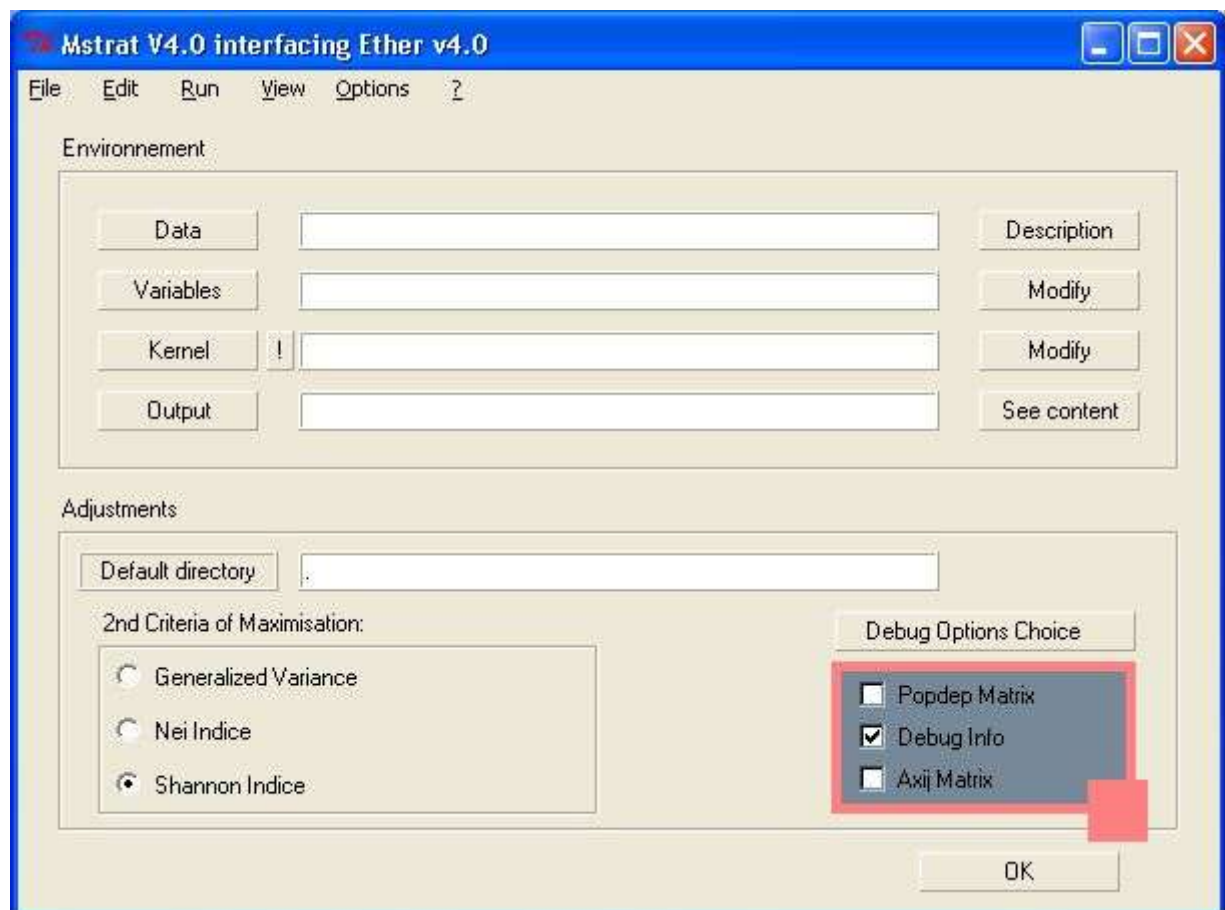
- Ether launching and input files working out to launch R.

These files use active qualitative variables (quali.txt) and active quantitative variables (quanti.txt). Tests are performed on the number of qualitative variables and the number of quantitative variables to determine if multivariable analysis could be done.

- R software runs in batch with the input file: quali.txt, quanti.txt and in.txt which contains commands necessary to launch R. Two files are created in output:
the file which gives the number of axes of principal component analysis (axe.txt) and the file giving the coordinates of accessions on these axis (out.txt). The out.txt file is in the vargenr directory. Other temporary files are stored in the vargenr directory and destroyed at the end of the process.
- Ether program is launched with the same command line than in the first point. Its goal is checking that quanti.txt, quali.txt and in.txt don't exist anymore and that out.txt does exist. It gives in output construction and redundancy files.

Multivariable analysis have to be performed on complete data file whereas allelic richness optimization can be done on incomplete data file. Complete and incomplete data files will be put into MStrat and be compared on the accession and individual columns. The variables files will have to be same-named and having the same number of variables. Blanks are taken into account in the comparison of the file.

b.Debugging



In the options menu you can select by clicking on "debug options choice" the different choices of debugging:

- Popdep matrix
- Debug Info
- Axij Matrix

Debug info will put all alert messages in the debug.txt file of the program directory and put the process tracing in an other debug.txt file which would be created in the default directory.

The other choices permit putting the used matrixes into files so you can check it in error cases

Beware! Only use it when necessary because it will make the process time much longer

c. Default options

To avoid re-entring settings values at each launching, your personal settings are saved in the default.mst file

The parameters of the environment of Tcl interface are kept in this file, located in the same directory as Mstrat.tcl.

The parameters are:

- the location of the input files
- the number of replicates
- the core size
- the default maximum value and
- the step for the definition of the core size

10. Technical

Data table is successively changed into 3 different matrixes used by MStrat:

a. Popdep matrix

Popdep is for "population de départ" which means in french: "start population".

This matrix contains:

- non-working variables coded with 0
- quantitative variables
- qualitative variables coded with their class number: each time a new value is found, a new class is created

Popdep matrix can also manage individuals. It has the same dimensions as the start table ones.

Example :

Data table

Accession	Individual	Quanti Var	Quali Var	Quali Var
1	1	326	p	0
2	1	451	pr	1
3	1	281	pc	1
3	2	422	pc	0

popdep matrix

Quanti Var	Quali Var	Quali Var
326	1	1
451	2	2
281	3	2
422	3	1

To split the X quantitative variable into 6 classes, we have to do this:

$$[\max(X) - \min(X)] / 6 = [451 - 281] / 6 = 28.33$$

So, The individual will be represented in the class n°2 because:

$$\text{sup round}([326 - 281] / 28.33) = \text{sup round}(1.58) = 2$$

Now, we can build up the Axij matrix

Notice: you can get this matrix by selecting it in the debug choices of the Option menu

b. Axij matrix

Axij matrix's goal is to work out if one variable class is represented in one accession.

Axij dimensions are:

- rows: accessions
- columns number: sum of classes number

We have to create classes from continuous distribution to permit the maximization algorithm process, because we want to maximize the number of classes in the core collection.

So, we split the values of equal-sized classes in which individuals are stored.

The Axij matrix contains the number of accessions for each variable class.

Example :

Axij

classe n°	1	2	3	4	5	6	1	2	3	1	2
accession n°1	0	1	0	0	0	0	1	0	0	1	0
accession n°2	0	0	0	0	0	1	0	1	0	0	1
accession n°3	1	0	0	0	1	0	0	0	1	1	1

Note: you can get this matrix by selecting it in the debug choices of the Option menu

c. AxijFreq matrix

The AxijFreq matrix contains the frequency distributions of all classes in each accession.

The process considers missing data.

Frequencies are calculated to work out same-weighted accession.

Here is an example:

Classe n°	1	2	3	4	5	6	1	2	3	1	2
Accession n°1	0	1	0	0	0	0	1	0	0	0.66	0
Accession n°2	0	0	0	1	0	1	0	1	0	0	0.66
Accession n°3	1	0	1	0	0	0	0	0	1	0.33	0.33

The AxijFreq matrix works out the average of all frequencies subcore accession from each class and each variable

Notice: AxijFreq is used by optimizing functions for Nei and Shannon indices.

d.Maximization algorithm

MStrat also uses an improved algorithm of maximization to optimize results reliability

The maximization process is based on the variables marker classes represented in a core collection. For each core collection or subcore collection, a score is calculated on marker variables as the number of represented classes. To do it as well, Axij matrix is used by counting values above or equals to 1 for the core collection accessions

- 1st step : random choice of one core collection of n accessions from data table N accessions
- 2nd step : creation of N subcore collections by dropping each accession, one by one.
- 3th step : a score is calculated for each subcore.
- 4th step : we drop one accession from the core collection where the subcore score is the highest because it's the accession which gives the less diversity. If 2 accessions are equals, a 2nd diversity criterion of these both two subcores is calculated. The accession which has the less one is dropped out.
- 5th step : we set up core collections of N accessions by adding each accession which is not already in the subcore
- 6th step : the score of each core collection is calculated. We add the accession which builds the core collection having the highest score on active marked variables

While the core collection score is improved, we repeat each step. To avoid perpetual loop (when the score is first incremented then decremented by 1 in each iteration), we define a breaking value for the loop iterations.

A.Common errors

File or directory doesn't exist

If you use addresses with spaces the program only looks for the first part and may not find the entire file address you asked for

Accession in double at line 3

two lines of the data file have the same number of accession and the same number of individual

Invalid value on line 1 / a variable 10 declared as quanti

the program has detected a non quantitative value for the variable number 10 at the line 1.

Error of lecture 8

means that an invalid value has been found at the line 8 (such as empty line).

Line 27 : error 1

at the line 27 one integer is missing for the number of accession or the number of the individual. The error can be located on the previous line in the data file or be due to a wrong variable data.

Invalid value for variable 9 class : 0

Zero has been written as the number of classes of the 9th variable which is a quantitative variable.

No active variable

No active variable has been indicated in the variable data.

Core size (10) higher than the total size (8)

The size of the core asked for is higher than the total number of accessions.

B.Limits of MStrat

Somes limits are implemented into the MStrat program

- Maximum number of alleles = 75
- Maximum number of classes = 1000
- Maximum length of the class name = 100 characters
- Maximum length of file directory = 128 characters
- Minimum number of accessions = 30