Valid Expressions

```
S →  HELP
     | PRINT
     | A
     | LIST
     | REMOVE
     | EXP
     | ELEMENT_NAMES
     | READ
     | WRITE
     | SAVE
```

Help

```
HELP →  ?CHAINE
        | help(CHAINE)
        | help(“CHAINE”)
        | help(“*”)
```

Basic arithmetic functions

```
FUNCTION →    mode(VAR)

              cat(VAR)

              length(VAR)

              log2(VAR) # logarithms base 2 of x

              log10(VAR) # logaritms base 10 of x

              exp(VAR) # Exponential of x

              cos(VAR) # Cosine of x

              sin(VAR) # Sine of x
```

```
tan(VAR) #Tangent of x
acos(VAR) # arc-cosine of x
asin(VAR) # arc-sine of x
atan(VAR) #arc-tangent of x
abs(VAR) # absolute value of x
sqrt(VAR) # square root of x
```

STAT_FUNCTION →

```
max(VAR)
min(VAR)
range(VAR)
length(VAR)
sum(VAR)
prod(VAR)
mean(VAR)
sd(VAR) # Standard deviation
var(VAR)
sort(VAR)
```

Assigning values to variables

VARS → VAR
| VAR,VAR

VAR → CHARACTER COMB
| ._COMB
| .CHARACTER COMB

COMB → . | _
| CHARACTER
| D

| COMB COMB
| *eps*

A → VAR ASSIGN EXP | RENAME ASSIGN VECTOR | LEVELS ASSIGN VECTOR | SUBSET_DATAFRAME ASSIGN VAR

ASSIGN → <- | =

EXP -> BASIC_TYPE
| VECTOR
| VAR
| FUNCTION
| STAT_FUNCTION
| TYPE
| TEST_TYPE
| CONVERT
| CHECK_NA
| CHECK_NAN
| SUBSET_VECTOR
| EXCLUDE_ELEMENT
| SELECT_ELEMENT
| CREATE_MATRIX
| TRANSPOSE
| DIMENSION
| SUBSET_MATRIX
| SELECT
| EXCLUDE
| SPEC_MATRIX_FUNCTION
| CREATE_FACTOR
| CHECK_FACTOR
| CONVERT_FACTOR
| INDIVID_PER_LEVEL
| SPEC_FACTOR_FUNC
| LEVELS

```
                | CREATE_DATAFRAME
                | CONVERT_DATAFRAME
                | SUBSET_DATAFRAME
                | SPEC_DATAFRAME_FUNCTION
                | SEQ
                | RSEQ
                | CREATE_LIST
                | SUBSET_LIST
```

PRINT → VAR | print(VAR)
LIST → ls()
REMOVE → rm(VARS)

Basic data types

BASIC_TYPE → LOGICAL
| NUMERIC
| STRING
| COMPLEX

COMPLEX → Di

LOGICAL → TRUE
| FALSE
| T
| F

NUMERIC → INTEGER | DOUBLE
INTEGER → DL
| DedL
| -DL
| -DedL
| +DL
| +DedL
d → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

D → dD
| d

DOUBLE → D | .D | D.D | D.Ded
| +D | +.D | +D.D | +D.Ded
| -D | -.D | -D.D | -D.Ded

STRING → “CHAINE”
| ‘CHAINE’

CHAINE → CHARACTER CHAINE
| CHAINE\’CHARACTER
| CHAINE\”CHARACTER
|CHARACTER

CHARACTER → a | b | c … | z | A | … | Z

TYPE → typeof(BASIC_TYPE)
| typeof(VAR) | class(VAR)

TEST_TYPE → is.numeric(VAR)
| is.character(VAR)
| is.logical(VAR) | is.complex(VAR)

CONVERT → as.numeric(VAR)
| as.character(VAR)
| as.logical(VAR)

/*
 * Conversion d’un string to numeric est possible : returns NA (not available)
 */

Vectors

VECTOR → c(CL)
| c(CN)
| c(CS)
| c(CL)

| c(CV)
| c(CS,CN,TL)
| c(CS,CN)
| c(TS,TL)
| c(CN,CL)
| c(CNAMED)

CNAMED → CNAMED_N
| CNAMED_L
| CNAMED _S

CNAMED_N → CHAINE = NA
| CHAINE = NUMERIC
| CHAINE = NUMERIC , CNAMED_N
| CHAINE = NA, CNAMED_N

CNAMED_L → CHAINE = NA
| CHAINE = LOGICAL
| CHAINE = LOGICAL , CNAMED_L
| CHAINE = NA, CNAMED_L

CNAMED_S → CHAINE = NA
| CHAINE = STRING
| CHAINE = STRING , CNAMED_S
| CHAINE = NA, CNAMED_S

CHECK_NA → is.na(VAR)
CHECK_NAN → is.nan(VAR)

CV → VECTOR,CV
| VECTOR
CL → LOGICAL,CL
| LOGICAL

CN → NUMERIC,CN
| NUMERIC
CS → STRING,CS
| STRING

ELEMENT_NAMES → names(VAR)

SUBSET_VECTOR → var[D]
| var[D:D]
| var[c(D,D)]
| var[STRING]

EXCLUDE_ELEMENT → var[-D]
| var[-c(D,D)]
| VAR [-(D:D)]

SELECT_ELEMENT → var[var LOG_OP BASIC_TYPE]
| var [!CHECK_NA]
| var [CHECK_NA]

LOG_OP → ==
| !=
| >=
| <=
| <
| >

Matrices

VECTORS → VECTOR
| VECTOR,VECTORS

CREATE_MATRIX → rbind(VARS)
| rbind(VECTORS)

| cbind(VARS)
| cbind(VECTORS)
# c for column and r for row
| matrix(data = VECTOR ,nrow = D , ncol = D , byrow = LOGICAL , dimnames = list(VECTORS))

RENAME → rownames(VAR)
| colnames(VAR)

TRANSPOSE → t(VAR)

DIMENSION → ncol(VAR)
| nrow(VAR)
| dim(VAR)

SUBSET_MATRIX → VAR[D,D]
| VAR[D,]
| VAR[D:D,]
| VAR[D:D,D:D]
| VAR[VECTOR,]
| VAR[,D]
| VAR[,D:D]
| VAR[,VECTOR]
| VAR[VECTOR,VECTOR]

SELECT → VAR[D,D]
| VAR[D,]
| VAR[,D]
| VAR[STRING,STRING]
| VAR[STRING,]
| VAR[,STRING] VAR[STRING,D]
| VAR[D,STRING]
| VAR[VAR LOG_OP BASIC_TYPE,]
| VAR[VAR LOG_OP BASIC_TYPE,VAR LOG_OP BASIC_TYPE]
| VAR[,VAR LOG_OP BASIC_TYPE]

EXCLUDE → VAR[-D,-D]
| VAR[-D,]
| VAR[,-D]

SPEC_MATRIX_FUNCTION → rowSums(VAR)
| colSums(VAR)
| colMeans(VAR)
|rowMeans(VAR)
| apply(VAR,1,STAT_FUNCTION)
|apply(VAR,2,STAT_FUNCTION)

Factors

CREATE_FACTOR → factor(VECTOR)
| factor(VAR,levels = VECTOR)
| factor(VAR)

CHECK_FACTOR → is.factor(VAR)

CONVERT_FACTOR → as.factor(VAR)

INDIVID_PER_LEVEL → summary(VAR)

LEVELS → levels(VAR)

SPEC_FACTOR_FUNC → tapply(VAR,VAR,STAT_FUNCTION)
| table(VAR)| table(VAR,VAR)

Data frames

CREATE_DATAFRAME → data.frame(COLS)

COLS → COL
| COL,COLS

COL → CHAINE=VECTOR
| CHAINE = VAR
| CHAINE = BASIC_TYPE

CHECK_DATAFRAME → is.data.frame(VAR)

CONVERT_DATAFRAME → as.data.frame(VAR)

/* You can use t() as same as Matrix to transpose a data frame*/

SUBSET_DATAFRAME → VAR$CHAINE
|VAR[,D]
| VAR[,STRING]
| VAR[,VECTOR]
| VAR[,-D]
| VAR$CHAINE LOG_OP BASIC_TYPE
| VAR[VAR$CHAINE LOG_OP BASIC_TYPE,]
| VAR[VAR$CHAINE LOG_OP BASIC_TYPE, VECTOR]
| VAR[VAR,VAR]
| subset(VAR, CHAINE LOG_OP BASIC_TYPE)
| attach(VAR),detach(VAR)

SPEC_DATAFRAME_FUNCTION (same as SPEC_MATRIX_FUNCTION)

Sequences

SEQ → seq(D,D,D.D)
| seq(SEQ_PARAM)
| rep(D,D)
| seq(D:D) | sequence(c(CN))

SEQ_PARAM → length=D
| label = c(CS)
| from = D
| to =D
| SEQ_PARAM,SEQ_PARAM

Random sequences :

RSEQ → PFUNC(DISTRIB_PARAMS)

P → r
| d
| p
| q

FUNC → norm | exp | gamma | nbinom | unif | geom | cauchy | pois | f | t | logis

DISTRIB_PARAMS → DISTRIB_PARAMS
| DISTRIB_PARAMS, DISTRIB_PARAMS
| D
| D.D
| scale =D
| location = D
| mean = D
| rate = D

Lists

CREATE_LIST → list(COLS)

/* element_names and length already exists */

SUBSET_LIST → VAR$CHAINE

| VAR[[STRING]]

| VAR[[D]]

| VAR[[D]][D]

Importing Data

READ → read.delim(file.choose(STRING))

| read.csv(file.choose(STRING))

| read.csv2(file.choose(STRING))

| read.tsv(file.choose(STRING))

Exporting Data

WRITE → data(STRING)

| write.table(VAR,PARAMS)

| write.csv(VAR,PARAMS)

| write.csv2(VAR,PARAMS)

SAVE → saveRDS(VAR,STRING)

| readRDS(STRING)

| load(STRING)

| save(VARS,file=STRING)

|save.image(file=STRING)

PARAMS → file = STRING

| sep = “SEP”

| row.names = LOGICAL

| col.names = LOGICAL

| row.names = NA

| col.names = NA

SEP → , | ; | \t