Valid Expressions

S → HELP
| PRINT
| LEVELS AEXP1
| LIST
| SUBSET_DATAFRAME AEXP2
| REMOVE
| ELEMENT_NAMES
| READ
| WRITE
| SAVE

V → VAR V'

V' → V1 | [V2]

V1 → ASSIGN EXP
| eps
| AFTER_VAR
| SL
| SD

V2 → -EE2
| SE2
| SM1
| S2
| EX2

AEXP1 → ASSIGN VECTOR | eps
AEXP2 → ASSIGN VAR | eps

Help

HELP → ?CHAINE | help(H1
H1 → CHAINE)
|“H2
H2 → CHAINE”) | *”)

Basic arithmetic operations

E → E E’ | VAR | NUMERIC
E’ → + T | - T | T
T → T T’ | F | VAR | NUMERIC
T’ → *F | /F | ^F | %% F
F → (E) | D

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 VAR2
VAR2 → eps|,VAR

VAR → CHARACTER COMB
| .VAR3
VAR3 → _COMP | CHARACTER COMB

COMB → . COMB1
| CHARACTER COMB1
| D COMB1
| *eps*

COMB1 → eps | COMB

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

ASSIGN → <- | =

EXP -> V
BASIC_TYPE
| VECTOR
| FUNCTION
| STAT_FUNCTION
| TYPE
| TEST_TYPE
| CONVERT
| CREATE_MATRIX
| TRANSPOSE
| DIMENSION
| SPEC_MATRIX_FUNCTION
| CREATE_FACTOR
| INDIVID_PER_LEVEL
| SPEC_FACTOR_FUNC
| LEVELS
| CREATE_DATAFRAME
| SUBSET_DATAFRAME
| SEQ
| RSEQ
| CREATE_LIST

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

Basic data types

BASIC_TYPE → LOGICAL
| NUMERIC
| STRING
| COMPLEX

COMPLEX → Di

LOGICAL → T LOGICAL3| F LOGICAL2

LOGICAL2→ eps | ALSE
LOGICAL3→ eps | RUE

NUMERIC → INTEGER | DOUBLE
INTEGER → D INTEGER2 | NEG_INT | POS_INT

INTEGER2 → L | e dL
NEG_INT → - D INTEGER2
POS_INT → +D INTEGER2
d → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
D → d D2
D2 → D | eps
DOUBLE → .D | D DOUBLE2
| POS_DOUBLE
|NEG_DOUBLE
DOUBLE2 → eps | .D | .Ded
POS_DOUBLE → +D DOUBLE2 | +.D
NEG_DOUBLE → -D DOUBLE2 | -.D

STRING → “CHAINE”
| ‘CHAINE’
CHAINE → CHARACTER CHAINE
| CHARACTER\ CHAINE2
|CHARACTER

CHAINE2 → eps | ’CHAINE | \”CHAINE

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

TYPE → typeof(TYPE2)
| class(VAR)
TYPE2 → BASIC_TYPE
| VAR

TEST_TYPE → is.IS
IS → numeric(VAR)
| character(VAR)
| logical(VAR)
| complex(VAR)
| na(VAR)
| nan(VAR)
| factor(VAR)
| dataframe(VAR)

CONVERT → as.AS

AS → numeric(VAR)
| character(VAR)
| logical(VAR)
| factor(VAR)
| dataframe(VAR)
/*
* Conversion d'un string to numeric est possible : returns NA (not available)
*/

Vectors

VECTOR → c(VECTOR2)
VECTOR2 → CL

| CN VECTOR 4
| CS VECTOR 3
| CV
| CNAMED

VECTOR3 → eps
| ,VECTOR33
VECTOR 33 → CN VECTOR333
| CL
VECTOR 333 → ,CL
| eps
VECTOR 4 → eps
| ,CL

CNAMED → CNAMED_N
| CNAMED_L
| CNAMED _S

CNAMED_N → CHAINE = CNAMED_N1
CNAMED_N1 → NA CNAMED_N2
| NUMERIC CNAMED_N3
CNAMED_N2 → eps
| ,CNAMED_N
CNAMED_N3 → eps
| ,CNAMED_N

CNAMED_L → CHAINE = CNAMED_L1
CNAMED_L1 → NA CNAMED_L2
| LOGICAL CNAMED_L3
CNAMED_L2 → eps
| ,CNAMED_L
CNAMED_L3 → eps
| ,CNAMED_L

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

CNAMED_S1 → NA CNAMED_S2
| STRING CNAMED_S3

CNAMED_S2 → eps
| ,CNAMED_S

CNAMED_S3 → eps
| ,CNAMED_S

CV → VECTOR CV2

CV2 → eps
| ,CV

CL → LOGICAL CL2

CL2 → eps
| ,CL

CN → NUMERIC CN2

CN2 → eps
| ,CN

CS → STRING CS2

CS2 → eps
| , CS

ELEMENT_NAMES → names(VAR)

AFTER_VAR → [AFTER_VAR2]

AFTER_VAR2 → D AFTER_VAR3
| c(D,D)
| STRING

AFTER_VAR3 → eps
| :D

EE2 → D
| c(D,D)
| (D:D)

SE2 → VAR LOG_OP BASIC_TYPE
| !CHECK_NA
| CHECK_NA

LOG_OP → ==
| !=
| >=
| <=

Matrices

VECTORS → VECTOR VECTORS2

VECTORS2 → eps | ,VECTORS

CREATE_MATRIX → rbind(RC)
| cbind(RC)
c for column and r for row
| matrix(data = VECTOR ,nrow = D , ncol = D , byrow = LOGICAL , dimnames = list(VECTORS))

RC → VARS | VECTORS

RENAME → rownames(VAR)
| colnames(VAR)

TRANSPOSE → t(VAR)

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

SM1 → SMD | SMP | SMV

SMP → , SMP2
SMP2 → D SMP3 | VECTOR
SMP3 → eps | :D
SMD → D, SMD2
SMD2 → ,D | :D SMD3| eps | :D,
SMD3 → eps | ,SMD4
SMD4 → eps | D:D
SMV → VECTOR SMV2
SMV2 → ,VECTOR | eps

S2 → S4 | S2 | S3 | S5

S3 → STRING S33| ,STRING |
S33 → , | ,STRING | ,D
S4 → D , S44
S44 → D|eps | STRING
S5 → VAR[S55]
S55 → VAR LOG_OP BASIC_TYPE, S555 | ,VAR LOG_OP BASIC_TYPE
S555 → eps | ,VAR LOG_OP_BASIC_TYPE

EX2 → -EX3| ,-D
EX3 → D EX33 |,D
EX33 → ,D |,

SPEC_MATRIX_FUNCTION → rowSums(VAR)
| colSums(VAR)
| colMeans(VAR)
|rowMeans(VAR)
| apply(VAR,APP)

APP → 1,STAT_FUNCTION
| 2,STAT_FUNCTION

Factors

CREATE_FACTOR → factor(FP)
FP → VECTOR VAR VAR,levels = VECTOR
FP 2 → eps | ,levels = VECTOR

INDIVID_PER_LEVEL → summary(VAR)

LEVELS → levels(VAR)

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

TP → VAR TP2
TP2 → eps | ,VAR

Data frames

CREATE_DATAFRAME → data.frame(COLS)

COLS → COL COLS2
COLS2 → eps | ,COLS

COL → CHAINE=COL2
COL2 → VECTOR | VAR | BASIC_TYPE

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

SUBSET_DATAFRAME → subset(VAR, CHAINE LOG_OP BASIC_TYPE)
| attach(VAR),detach(VAR)
SD → $SD1 | [SD2]
SD2 → ,SDP
| VAR SD22
SD22 → ,VAR | $CHAINE LOG_OP BASIC_TYPE, SD222
SD222 → eps | VECTOR | VAR,VAR
SDP → D | STRING | VECTOR | -D
SD1 → CHAINE SD11
SD11 → eps | LOG_OP BASIC_TYPE

Sequences

SEQ → | seq(SEQP)
| rep(D,D)
| sequence(c(CN))

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

SEQD → :D | ,D,D.D
SEQR → eps

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 → | D.D DPD
| scale =D
| location = D
| mean = D
| rate = D

DPD → ,DISTRIB_PARAMS | eps | .D

Lists

CREATE_LIST → list(COLS)

/* element_names and length already exists */

SL → $CHAINE | [[SL2]

SL2 → STRING]] | D SLD

SLD →]] SLD2

SLD 2 → eps | [D

Importing Data

READ → read.READ2

READ2 → delim(file.choose(STRING)) |csv(file.choose(STRING)) | csv2(file.choose(STRING))

Exporting Data

WRITE → data(STRING)

| write WRITE2

WRITE2 → table(VAR,PARAMS) | csv(VAR,PARAMS) | csv2(VAR,PARAMS)

SAVE → saveRDS(VAR,STRING)

| readRDS(STRING)

| load(STRING)

| save SAVE2

SAVE2 → (VARS,file=STRING) | .image(file=STRING)

PARAMS → file = STRING

| sep = “SEP”

| row.names = RCN

| col.names = RCN

RCN→ LOGICAL | NA

SEP → , | ; | \t