LES OPERATEURS SAS

Symbole	***Equivalence***	***Définition***
**		Exponentielle
><	MIN	Minimum
<>	MAX	Maximum
*		Multiplication
/		Division
-		Soustraction
+		Addition
=	EQ	Egal à
^=	NE	Différent de
>	GT	Plus grand que
<	LT	Plus petit que
<=	=<	Inférieur ou égal à
>=	=>	Supérieur ou égal à
	IN	Egal à une liste
&	AND	Et
|	OR	Ou
^	NOT	Négation
||		Concaténation

LES FONCTIONS

Les fonctions arithmétiques

ABS(num) donne la valeur absolue (= la valeur positive) d'un nombre

ABS(5.8) => 5.8 ou ABS(-58) => 58

DIM(variable) donne le nombre d'éléments qu'il y a dans un ARRAY

si on a défini ARRAY temp (*) x1-x5, alors DIM(temp) => 5

MAX(num_1, ..., num_n) calcule la valeur maximale de la liste de chiffres ou de variables

MAX(1.5, 1.8, 1.79999) => 1.8

MIN(num_1, ..., num_n) calcule la valeur minimale de la liste de chiffres ou de variables

MIN(1.5, 1.8, 1.79999) => 1.5

MOD(num_1, num_2) calcule le reste de la division de num_1 par num_2

MOD(10, 3) => 1

SQRT(num) calcule la racine carré d'un nombre. Pour d'autres racines, on procèdera avec un exposant fractionnaire comme x = y ** (1/3) pour obtenir une racine cubique.

SQRT(9) => 3

Les fonctions de troncation de nombre

CEIL(num) arrondi à l'entier supérieur

CEIL(5.8) => 6

FLOOR(num) arrondi à l'entier inférieur

FLOOR(5.8) => 5

INT(num) tronque à l'entier

INT(5.8) => 5

ROUND(num, nombre) arrondi le nombre num en fonction du nombre d'unités d'arrondi

ROUND(5.8, 1) => 6
ROUND(5.8) => 6
ROUND(5.7999, 0.1) => 5.8
ROUND(57899.1,10) => 57900

Les fonctions mathématiques

EXP(num) calcule l'exposant de num

EXP(1) => 2.7182818285
EXP(0) => 1
EXP(3.123) => 22.714420793

LOG(num) calcul le logarithme népérien de num

LOG(1) => 0
LOG(0) => . et cela génére un message d'erreur
LOG (0.5) => -0.693147181
LOG(12) => 2.4849066498
LOG(EXP(1)) => 1

LOG2(num) calcul le logarithme en base 2 de num

LOG2(1) => 0
LOG2(0) => . et cela génére un message d'erreur
LOG2 (0.5) => -1
LOG2(12) => 3.5849625007
LOG2(2) => 1

LOG10(num) calcul le logarithme en base 10 de num

LOG10(1) => 0
LOG10(0) => . et cela génére un message d'erreur
LOG10(0.5) =>-0.301029996
LOG10(12) => 1.079181246
LOG10(2) => 1

Les fonctions trigonométriques

COS(num) calcule le cosinus de num

si pi = 3.14159265358979

```
COS(  0 / 180 * pi) =>  1.000 pour 0 degré
COS( 45 / 180 * pi) =>  0.707 pour 45 degrés
COS( 90 / 180 * pi) =>  0.000 pour 90 degrés
COS(180 / 180 * pi) => -1.000 pour 180 degrés
COS(270 / 180 * pi) =>  0.000 pour 270 degrés
COS(360 / 180 * pi) =>  1.000 pour 360 degrés
```

SIN(num) calcule le sinus de num

si pi = 3.14159265358979

```
SIN(  0 / 180 * pi) =>  0.000 pour 0 degré
SIN( 45 / 180 * pi) =>  0.707 pour 45 degrés
SIN( 90 / 180 * pi) =>  1.000 pour 90 degrés
SIN(180 / 180 * pi) =>  0.000 pour 180 degrés
SIN(270 / 180 * pi) => -1.000 pour 270 degrés
SIN(360 / 180 * pi) =>  0.000 pour 360 degrés
```

TAN(num) calcule la tangente de num

ARCOS(num) calcule l'arcosinus de num (qui est un nombre entre 0 et 1). C

ARSIN(num) calcule l'arcsinus de num

ARTAN(num) calcule l'arctangente de num

Les fonctions statistiques

N(num_1, ..., num_n) calcule la fréquence de valeurs non-manquantes

N(1, 2, 4, 8, 10, 8, 4, 2, 1) => 9

NMISS(num_1, ..., num_n) calcule la fréquence des valeurs manquantes

NMISS(1, 2, 4, 8, 10, 8, 4, 2, 1) => 0

SUM(num_1, ..., num_n) calcule la somme de la liste des arguments non-manquants

SUM(1, 2, 4, 8, 10, 8, 4, 2, 1) => 40

MEAN(num_1, ..., num_n) calcule la moyenne de la liste des arguments non-manquants

MEAN(1, 2, 4, 8, 10, 8, 4, 2, 1) => 4.44

RANGE(num_1, ..., num_n) calcule l'écart entre les valeurs minimale et maximale de la liste des arguments non-manquants

RANGE(1, 2, 4, 8, 10, 8, 4, 2, 1) => 9

STD(num_1, ..., num_n) calcule l'écart-type de la liste des arguments non-manquants

STD(1, 2, 4, 8, 10, 8, 4, 2, 1) => 3.40

STDERR(num_1, ..., num_n) calcule l'erreur-standard à la moyenne de la liste des arguments non-manquants

STDERR(1, 2, 4, 8, 10, 8, 4, 2, 1) => 1.13

VAR(num_1, ..., num_n) calcule la variance de la liste des arguments non-manquants

VAR(1, 2, 4, 8, 10, 8, 4, 2, 1) => 11.53

CV(num_1, ..., num_n) calcule le coefficient de variation de la liste des arguments non-manquants

CV(1, 2, 4, 8, 10, 8, 4, 2, 1) => 76.39

CSS(num_1, ..., num_n) calcule la somme corrigée des carrés de la liste des arguments

non-manquants

CSS(1, 2, 4, 8, 10, 8, 4, 2, 1) => 92.22

Les fonctions probabilistiques

PROBNORM(x) calcule la probabilité qu'une observation d'une distribution normale standardisée (avec une moyenne = 0 et écart-type = 1) soit plus petite ou égale à x.

PROBNORM(0) => 0.500 soit à la moyenne, la probabilité est de 50%
PROBNORM(1.96) => 0.9750 soit à la valeur de 1.96, à pratiquement 2 écart-types, la probabilité est de 97,5%

PROBT(x, dld) calcule la probabilité qu'une observation d'une distribution de t de Student, avec ddl degré de liberté soit plus petite ou égale à x. Si on veut obtenir le niveau de signification d'un test de t bilateral, on doit faire p = (1 - PROBT(ABS(x),ddl))*2. Dans le cas d'un test unilatéral, on fera p = 1 - PROBT(ABS(x),ddl). On peut ainsi reconstruire un tableau de probabilité complet pour la v.a. de Student.

PROBT(1.45, 10) => 0.9112. La probabilité unilatérale est égale à 8,88 %; si le test est bilatéral, la probabilité est égale à 17,72 %
PROBT(1.96, 1000) => 0.9748. La probabilité unilatérale est égale à 2,52 %; si le test est bilatéral, la probabilité est égale à 5,04 %

PROBF(x, ddln, ddld) calcule la probabilité qu'une observation d'une distribution de Fisher, avec ddln degré de liberté au numérateur (effet factoriel) et ddld degré de liberté au dénominateur (effet résiduel) soit plus petite ou égale à x. Si on veut obtenir le niveau de signification d'un test de F, on doit faire p = 1 - PROBF(x, ddln, ddld).

PROBF(3.32, 2, 3) => 0.8264. La probabilité est égale à 17,36 %

PROBCHI(x, ddl) calcule la probabilité qu'une observation d'une distribution de Chicarré, avec ddl degré de liberté soit plus petite ou égale à x. Si on veut obtenir le niveau de signification d'un test de Chicarré, on doit faire p = 1 - PROBCHI(x, ddl).

POISSON(x, n) calcule la probabilité qu'une observation d'une distribution de Poisson de moyenne x, soit plus petite ou égale à n.

POISSON(1, 2) => 0.9197

PROBBNML(p, n, m) calcule la probabilité qu'une observation d'une distribution binomiale, avec une probabilité de succès p, un nombre d'essais n et un nombre de succès m, est plus petite ou égale à m.

PROBBNML(0.5, 10, 4) => 0.3770

Les 3 fonctions suivantes font l'inverse des précédentes. Elles donnent les valeurs des tables qui correspondent à une probabilité et un certain nombre de degré de liberté.

TINV (p, ddl) calcule une valeur d'une table de t de Student correspondante à une probabilité p et ddl degré de liberté.

TINV(0.95, 2) => 2.9199

FINV (p, ddln, ddld) calcule une valeur d'une table de F correspondante à une probabilité p et ddln degré de liberté au numérateur et ddld degré de liberté au dénominateur.

FINV(0.95, 2, 10) => 4.1028

CINV (p, ddl) calcule une valeur d'une table de Chicarré correspondante à une probabilité p et ddl degré de liberté.

CINV(0.95, 3) => 7.81

Les fonctions de génération de nombres "aléatoires"

UNIFORM(n) génére un nombre pseudo-aléatoire compris entre 0 et 1 [UNIFORM (n) est synonyme de RANUNI (n)]

NORMAL(n) génére un nombre dont la fréquence suit une distribution normale standardisée (avec une moyenne = 0 et écart-type = 1) [NORMAL (n) est synonyme de RANNOR (n)].

Pour générer une série de valeurs x correspondant à une variable aléatoire normale de moyenne mean et un écart-type std, on utilisera la formule x = mean + std * NORMAL(0)

RANPOI(n, lambda) génére un nombre dont la fréquence suit une distribution de Poisson de moyenne lambda.

RANBIN(n, nbr, p) génére un nombre dont la fréquence suit une distribution binomiale(nbr, p).

Les fonctions alphabétiques

COMPRESS(anum) enlève les blancs dans la chaîne de caractères anum. Si elle est utilisée avec deux arguments comme COMPRESS(anum, "%"), elle enlève les "%" dans la chaîne de caractères anum.

COMPRESS("C'est la vie") => "c'estlavie"
COMPRESS("C'est la vie", "e") => "c'st la vi"

LEFT(anum) aligne à gauche dans la chaîne de caractères anum.

LEFT(" C'est la vie ") => "c'est la vie "

RIGHT(anum) aligne à droite dans la chaîne de caractères anum.

LEFT(" C'est la vie ") => " c'est la vie"

TRIM(anum) enlève les blancs à droite (à la fin) dans la chaîne de caractères anum.

TRIM(" C'est la vie ") => " c'est la vie"

TRANSLATE(anum, t_1, f_1, ..., t_n, f_n) traduit dans la chaîne de caractère anum les caractères t par le caractère f correspondant.

```
TRANSLATE("    C'est la vie    ", " ", "*") => "****C'est la
vie****"
```

UPCASE(anum) transforme tous les caractères de la chaîne de caractère anum en majuscules.

```
UPCASE("C'est la vie") => "C'EST LA VIE"
```

LOWCASE(anum) transforme tous les caractères de la chaîne de caractère anum en minuscules.

```
LOWCASE("C'est la vie") => "c'est la vie"
```

REVERSE(anum) renverse lo'rdre de tous les caractères de la chaîne de caractère anum.

```
REVERSE("C'est la vie") => "eiv al tse'C"
```

LENGTH(anum) donne la longueur de la chaîne de caractère anum en ignorant les caractères blancs à la fin de la chaîne.

```
LENGTH("C'est la vie    ") => 12
```

INDEX(anum, string) cherche dans la chaîne de caractère anum la position de la chaîne ou du caractère string.

```
INDEX("C'est la vie    ", "v") => 10
INDEX("C'est la vie    ", "z") => 0
```

INDEXC(anum, t_1, ..., t_n) cherche dans la chaîne de caractère anum la position d'un des caractères t.

```
INDEXC("C'est la vie    ", "t", "l", "z") => 5
```

SUBSTR(anum, n, m) extrait dans la chaîne de caractère anum m caractères en commençant à la position n.

```
SUBSTR("C'est la vie", 7, 2) => "la"
```

SCAN(anum, n) extrait dans la chaîne de caractère anum le nième mot séparé par des blancs.

```
SCAN("C'est la vie", 2) => "la"
```

SCAN(anum, n, car) extrait dans la chaîne de caractère anum le nième mot séparé par le délimiteur car.

```
SCAN("C'est_la_vie", 2, "_") => "la"
```

Les fonctions spéciales

LAG n (anum) restitue la valeur qu'avait l'argument à la n ième ligne précédente.

DIF n (arg) fournit la valeur de la différence entre l'argument et la valeur qu'avait l'argument à la n ième ligne précédente.

PUT (variable,format.) permet de créer une variable caractère à partir d'une autre variable en indiquant le format désiré.

INPUT (variable,informat.) permet de créer une variable (le plus souvent numérique) à partir d'une variable caractère en spécifiant le informat désiré.

Les fonctions chronologiques

DATE() ou TODAY() donne la date du jour.

```
date1 = TODAY() => date1 =  15763 si on est le 27
février 2003.
```

TIME() donne l'heure du jour.

```
heure1 = time() => heure1 =  46024.94 si il est 12
h 47' 4.9410"
```

MDY(mois, jour, annee) transforme les informations d'une date stockée en trois variables en une nouvelle variable de type "date".

```
date1 = MDY( 8, 18, 1961) => date1 =   595.
```

DAY(date), MONTH (date) et YEAR(date) transforme les informations de date stockée en une variable de type "date" en trois nouvelles variables.

```
date1 = TODAY() => date1 =  15763 si on est le 27
février 2003.
jour = DAY(date1) => jour = 27
mois = MONTH(date1) => mois = 2
annee = YEAR (date1) => annee = 2003
```

HMS(heure, minute, seconde) transforme les informations d'une heure stockée en trois variables en une nouvelle variable de type "heure".

```
heure1 = HMS( 12, 18, 10.01) => heure1 =   44290.01
```

HOUR(heure), MINUTE (heure) et SECOND(heure) transforme les informations d'heure stockée en une variable de type "heure" en trois nouvelles variables.

```
heure1 = TIME() => heure1 =  46040.07 si il est 12
h 47' 20,07"
heure = HOUR(heure1) => heure = 12
jour = MINUTE(heure1) => minute = 47
seconde = SECOND(heure1) => seconde = 20.07
```

Les fonctions financières

MORT(Montant,Versement,Taux,Nombre):

- Montant = Quantité dûe
- Versement = Quantité remboursée à chaque versement
- Taux = Taux d'intérêt par période, il s'exprime sous forme de fraction (10/12)
- Nombre = Nombre de périodes de versement

La fonction, à partir des valeurs founies de 3 arguments, calcule le 4ième.

Les fonctions date

Fonctions	***Descriptions***
DATE()	Retourne la date courante
DATDIF(date_début,date_fin,base)	Retourne le nombre de jours entre 2 dates. Base est égale à '30/360' (base de 30 jours par mois et 360 jours sur l'année) ou 'ACT/ACT' (nombe de jours réels).
DATEJUL(julian-date)	Convertit une date au format yyddd (JULIAN) en date SAS (nombre de jours depuis le 01/01/1960).
DATEPART(datetime)	Extrait la partie date d'une date heure.
DAY(date)	Retourne le quantième du mois.
DHMS(date,hour,minute,second)	Retourne une date heure SAS à partir des éléments date, heure, minute et seconde (renvoie le nombre de secondes depuis le 01/01/1960)
HMS	Retourne une heure à partir des éléments heure, minute et seconde.
HOUR(time|datetime)	Retourne l'élément heure d'une date heure ou d'une heure.
INTCK('interval',from,to)	Retourne la différence entre 2 éléments temporels (date, heure, date-heure) exprimée en fonction de l''interval' défini (day, weekday, week, semimonth, month, qtr, semiyear, year,hour,minute,second...).
INTNX('interval',from,incrément)	Ajoute à un élément temporel (date, heure, date-heure) un incrémentcorrespondant au type d''interval' défini (day, weekday, week, semimonth, month, qtr, semiyear, year,hour,minute,second...).
JULDATE(date)	Convertit la date en julian date.
MDY(month,day,year)	Retourne une dat à partir des éléments mois, jour, année.
MINUTE(time|datetime)	Retourne les minutes d'une heure ou d'une date-heure.
MONTH(date)	Retourne le mois une date.
QTR(date)	Retourne le trimestre d'une date.
SECOND(time|datetime)	Retourne les secondes d'une heure ou date-heure.
TIME()	Retourne l'heure de la date du jour.