Quelques fonctions et routines méconnues ou peu utilisées en SAS®

```
  \/\|/\/
 \\  = =  //
  (  @ @  )
oooo (_) oooo
```

Est-ce que ça pourrait m'être utile?

```
      oooo
oooo  (    )
(    )  ) /
 \ (   ( /
  \ )   -
   -
```

Alain Voyer
utilisateur SAS® paresseux
Club des utilisateurs SAS de Québec, 8 mai 2019

Proposition

- Introduction
- Combien de fonctions?
- Familles de fonctions
- Duo et trio de fonctions
- Autres fonctions
- | | vs PUT vs CAT
- Informations sur les formats et dictionnaire
- Rappel: Code VS Fonction
- Conclusion
- Questions ?

Introduction

Au gré des versions SAS de nouvelles fonctions et routines apparaîssent ou des fonctions et routines existantes sont bonifiées par de nouvelles possibilités.

Mais ces nombreuses nouveautés peuvent passer inaperçues.

Un premier pas consiste à faire le tour de certaines des ces fonctions et routines pour en connaître l'existence.

Note: une **fonction** retourne une valeur et une **routine** peut retourner une ou plusieurs valeurs et on la repère avec le mot **CALL**

Même si on ne trouve pas immédiatement dans quelles circonstances on peut en faire une utilisation concrète, ces fonctions et routines peuvent faire leur chemin dans notre subconscient et devenir utiles dans un traitement que nous aurons à faire.

$f(x)$

Pour le bénéfice de la présentation, quand on évoque les fonctions on sous-entend les routines également

Combien de fonctions?

Comment calculer le nombre de fonctions de SAS version 9.4M6 (University Edition):

```
proc sql;
  select count(distinct(fncname))
    from dictionary.functions;
quit;
```

Résultat: 997 $f(x)$

4

Explorer le dictionnaire des fonctions pour trouver le nombre de fonctions.

Cette table du dictionnaire contient plusieurs champs dont l'explication est introuvable, du moins, pour le moment.

Avec la version 9.4M6 (University Edition) Il y a 1087 entrées au total et 997 si on enlève les doublons sur le nom de la fonction (ou routine)

Famille CAT

Fonctions: CAT, CATQ, CATS, CATT, CATX

Routines: CALL CATS, CALL CATT, CALL CATX

pour <u>concaténer</u> des chaînes de caractères...

mais <u>pas seulement</u> des chaînes...

Syntaxe générale de deux des fonctions:

```
c=cat(item1 <,…,itemN>);
c=catx(delim, item1 <,…,itemN>);
```

catt = cat mais supprime les espaces de fin (trim)

cats = cat mais supprime les espaces de début et de fin (strip)

catx = cats mais avec l'ajout d'un délimiteur

Les fonctions CAT, CATS, CATT sont assez semblables
(ont la même façon d'exprimer les arguments)

CATQ offre la possibilité d'ajouter des apostrophes ou
des guillements aux chaînes concaténées.

Famille CAT

Exemples de correspondance entre des fonctions CAT et l'opérateur de concaténation || combiné aux fonctions TRIM, LEFT qui sont apparues avant les CAT

Fonction	Opérateur
cat(a,b," $ ",d,e,f)	a || b || " $ " || d || e || f
catt(of b:)	trim(bus) || trim(boulier) || trim(bar) || trim(brac)...etc
cats(of c1-c4,g9)	trim(left(c1)) || trim(left(c2)) || trim(left(c3)) || trim(left(g9))
catx("/", of d1-d3)	trim(left(d1)) || "/" || trim(left(d2)) || "/" || trim(left(d3))

Le tableau présente la correspondance entre les fonctions CAT et l'opérateur de concaténation

Les fonctions TRIM et LEFT proviennent de la période "jurassique" de SAS
J'appelle ça la technique du "trimleftage" qui existe en abondance dans les traitements "patrimoniaux"

Le "rimleftage" offre un visuel abondamment garni de parenthèses et de barres verticales.
L'intérêt des fonctions CAT est de pouvoir utiliser le OF pour abréger la liste.

On ne peut pas faire ça avec l'opérateur de concaténation.

Famille CAT

```
c="Lotus";
n=123;
fc=cat(c,n);
oc=c||n;
lfc=lengthm(fc);
loc=lengthm(oc);
```

Quelques remarques:

- Fonction pas = opérateur
- Fonction convertit numérique
- Enlève les espaces
- Opérateur provoque Note
- Longueurs différentes

	FC	OC	LFC	LOC
1	Lotus123	Lotus 123	200	17

```
NOTE: Numeric values have been converted to character
      values at the places given by: (Line):(Column).
```

7

Une fonction ne se comporte pas exactement comme un opérateur notamment concernant la longueur résultante.

La routine donnera habituellement une longueur de 200 (le maximum par défaut)

Un opérateur donnera la longueur équivalente à la somme des longueurs de chaque argument.

Famille CAT

Exemple spécifique pour CATQ

```
n=123;
c="Lotus";
fc=catq("2ac",c,n);
oc='"'||c||'"'||","||'"'||n||'"';
```

	N	C	FC	OC
1	123	Lotus	"Lotus","123"	"Lotus"," 123"

```
NOTE: Numeric values have been converted to character
      values at the places given by: (Line):(Column).
```


CATQ permet l'utilisation de modificateurs. Ici on a 3 modificateurs.
Contrairement à d'autres fonctions (SCAN, COMPRESS,
FIND, COUNT, COMPARE) les modificateurs se placent au début

Le 2 signifie que l'on désire des guillements autour des variables
La lettre "a" demande d'ajouter ces guillemets à tous les arguments
La lettre "c" demande d'utiliser la virgule (comma) comme délimiteur.

Avec l'opérateur de concaténation, on obtient pas tout à fait
l'équivalent qu'avec la fonction CATQ.

On obtient également une note de conversion au journal.

Familles ANY et NOT

13 fonctions ANY

ANYALPHA: détecter la présence de caractère alphabétique

ANYDIGIT: détecter la présence de caractère numérique

Autres fonctions: ANYLOWER, ANYUPPER, ANYSPACE...

13 fonctions NOT

NOTALPHA: détecter la présence de caractère non alphabétique

NOTDIGIT: détecter la présence de caractère non numérique

Autres fonctions: NOTLOWER, NOTUPPER, NOTSPACE...

Fonctions ANY et NOT sont en complémentarité l'une par rapport à l'autre. ANYALPHA et NOTALPHA, etc.

Familles ANY et NOT

```
aa=anyalpha("418-Québec");
ad=anydigit("6Urg 911",2);

na=notalpha("G1K 7E2",-6);
nd=notdigit("911");
```

	AA	AD	NA	ND
1	5	6	5	0

	CAA	CAD	CNA	CND
1	Q	9	7	

CAA représente le caractère trouvé à la position AA, CAD pour la position AD, CNA pour NA et CND pour ND.

10

Quand le deuxième argument est absent ou positif
on fait la recherche de gauche à droite

Quand le deuxième argument est négatif
on fait la recherche de droite à gauche à partir de la position 6.

On retrouve donc le caractère 7 qui est en 5ème position

Famille SORT

Fonction SORT

Routines CALL SORTC, CALL SORTN

```
array x(3) $2 ("TT" "AA" "JJ");
call sortc(of x(*));
```

	X1	X2	X3
1	TT	AA	JJ
2	AA	JJ	TT

avant le tri

après le tri

Note: la fonction SORT retoune 1 si le tri a bien fonctionné et 0 autrement
Cette fonction est assez récente cependant.

11

Pratique pour trier des vecteurs.

La fonction SORT est documentée mais n'existe pas
à la version SAS 9.4 M3 mais existe en version 9.4M6

Elle retourne la valeur 1 si le tri à fonctionné et 0 autrement

La routine CALL SORTC est illustrée ici et son
utilisation est simple

Famille SORT

Découverte dans le dictionnaire... 9.4M6

Obs.	source	fncname	minarg	maxarg	fnctype	fncargs	fncprod
1	U	SORT	1	0	B	1073741823	B
2	U	SORT	1	0	B	1073741823	B
3	U	SORT	1	0	N	1073741823	B
4	U	SORT	1	0	N	1073741823	B
5	U	SORTC	1	0	B	1073741823	B
6	U	SORTC	1	0	B	1073741823	B
7	U	SORTC	1	0	N	1073741823	B
8	U	SORTC	1	0	N	1073741823	B
9	U	SORTCX	1	0	B	1073741823	B
10	U	SORTCX	1	0	N	1073741823	B
11	B	SORTH	1	0	B	63	B
12	U	SORTKEY	1	9	C	38347922	B
13	U	SORTKEYT	1	9	C	38347922	B
14	U	SORTN	1	0	B	1073741823	B
15	U	SORTN	1	0	B	1073741823	B
16	U	SORTN	1	0	N	1073741823	B
17	U	SORTN	1	0	N	1073741823	B
18	U	SORTNX	1	0	B	1073741823	B
19	U	SORTNX	1	0	N	1073741823	B

Liste des fonctions/routines SORTx selon le dictionnaire SAS des fonctions

Des doublons et plus...

Des fonctions/routines non documentées et/ou expérimentales ???

Les informations détaillées concernant les champs sont... introuvables.

Aidez-moi!

12

Le dictionnaire des fonctions s'est enrichi entre SAS 9.4M3 et 9.4M6.

Il semble exister 19 variantes de fonctions et routines consacrées au tri...

Il en reste 12 si on enlève les doublons d'observations
et 8 si on enlève les doublons sur le nom de la fonction/routine

Mais on ne sait pas exactement ce que veulent dire les champs.

Duo CHOOSE

CHOOSEC, CHOOSEN

retourne une valeur caractère ou numérique basée sur un index (positif –> gauche à droite ou négatif –> droite à gauche)

```
c=choosec(2, "CC","AA","BB");
d=choosec(5, "FF","EE","DD");
m=choosen(3, 1920,1900,1926);
n=choosen(-4, 1920,1900,1926);
```

	C	D	M	N
1	AA		1926	.

13

Choosec retourne la variable caractère selon un index (premier argument):
Syntaxe: CHOOSEC(index-expression, selection-1 <, …selection-n>)

Choosen retourne une variable numérique
La liste des éléments doit être spécifiée explicitement. On ne peut pas utiliser OF.

Quizz: Quelle est la longueur de la variable caractère C et D ? 2 ou 200
Réponse: 200

Duo WHICH

WHICHC, WHICHN

Recherche une valeur caractère ou numérique égale au premier argument et renvoie la position de la première valeur correspondante

```
b=whichc("BB", "CC","AA","BB");
z=whichc("ZZ", "FF","EE","DD");
n=whichn(1900, 1920,1900,1926);
o=whichn(1950, 1920,1900,1926);
```

	B	Z	N	O
1	3	0	2	0

whichc et whichn retournent toujours une valeur numérique.

0 signifie: pas trouvé

Combiné CHOOSEC-WHICHC

```
data;
  input Cote $1.;
  Pos     = whichc(Cote, "A"        ,"B"       ,"C"  ,"D"          ,"E"    );
  Verdict = choosec(Pos, "Excellent","Très bon","Bon","Insuffisant","Échec");
datalines;
B
E
C
F
A
;
run;
```

Ce "combiné" permet de transposer des valeurs.
On transpose ici une cote en un verdict.
Ainsi une cote B deviendra un verdict "Très bon", etc.

	COTE	POS	VERDICT
1	B	2	Très bon
2	E	5	Échec
3	C	3	Bon
4	F	0	
5	A	1	Excellent

Combiné permettant de faire une transposition de valeurs.
1 valeur pour une autre.

On transforme une cote en un verdict

Quelles sont les autres alternatives à ce combiné:
proc format, if then else if... , select when, array/do/index.

Trio FIND

Pour effectuer une recherche de caractère(s) ou sous-chaîne dans une chaîne à partir d'une position (optionelle) de départ

```
x=FIND(ch,sch<,dép><,mod>);
x=FINDC(ch,lst-c,<dép>,<mod>);
x=FINDW(ch,mot,dép<,car<,mod>>);
```

Argument "mod(ificateur.s)" pour changer le comportement de la fonction:

I: Ignore la casse, T: élimine les espaces de fin de chaîne

Fonctions FINDC pour trouver la présence d'un caractère dans une chaîne et FINDW pour chercher un "mot"

16

Les fonctions FIND ressemblent aux fonctions INDEX
mais avec plus de possibilités notamment une position de départ et des modificateurs (19)

Syntaxe: INDEX(source, excerpt),
INDEXC(source, excerpt-1 <, …excerpt-n>),
INDEXW(source, excerpt <, delimiters>)

19 modificateurs disponibles dont:
a or A**:** adds alphabetic characters to the list of characters.
d or D**:** adds digits to the list of characters.
i or I: ignores character case during the search.
l or L: adds lowercase letters to the list of characters.
t or T: trims trailing blanks from the string and character-list arguments.
u or U: adds uppercase letters to the list of characters.

Trio FIND

```
*  ----+---10----+---20----+---30----+-;
c="Club/des+utilisateurs/SAS+de/Québec";
f =find (c, "uB", "i");
fc=findc(c, "kÉ", "id");
fw=findw(c, "SAS","/+", "e");
```

	C	F	FC	FW
1	Club/des+utilisateurs/SAS+de/Québec	3	32	4

Fonctions connexes: famille INDEX, familles ANY et NOT, VERIFY

17

Les 3 fonctions de recherche dans une chaîne de caractères.
find: trouver la sous-chaine "UB" dans la chaîne sans tenir
compte de la casse (modificateur "i") et donner la position dans la variable f

findc: trouver la lettre "k" ou "É" en ignorant la casse et en ajoutant
les chiffres grâce au modificateur "d" et donner la position dans la variable fc
findw: trouver le mot "SAS" en utilisant comme délimiteur de mot les
caractères "/" et "+" et donner la position du mot dans la variable fw et
non pas du premier caractère de la chaine "SAS";

Modificateurs:
e or E counts the words that are scanned until the specified word is found,
instead of determining the character position of the specified word in
the string. Fragments of a word are not counted.

i or I ignores the case of the characters.

d or D adds digits to the list of characters.

Le + petit, le + grand

```
array a(*) a1-a4 (1920,1900,1926,.);
s=smallest(2, of a(*));
o=ordinal (2, of a(*));
l=largest (5, of a(*));
```

	S	O	L
1	1920	1900	.

smallest/largest ignorent les valeurs manquantes mais pas ordinal

18

Les fonctions SMALLEST, LARGEST et ORDINAL nous permettent de trouver le n ième plus petit ou plus grand élément d'une liste

Les fonctions MAX et MIN sont aussi disponibles mais ne permettent pas de trouver le x ième plus grand ou plus petit.

REVERSE

On connaît l'opérateur **=:** pour trouver ce qui **commence** par ...
Mais comment trouver une chaîne de caractère qui se **termine** par ...

```
c="Club SAS de Québec";
if reverse(c) =: reverse("Québec")
   then r="Trouvé!";
   else r="Zut!";
```

	C	R
1	Club SAS de Québec	Trouvé!

19

On peut utiliser:
if reverse(strip(c) =: "cebéuQ" ....

mais c'est plus compréhensible d'utiliser:
if reverse(strip(c) =: reverse("Québec");

COALESCE

Scénario: On désire connaître la date la plus récente de participation d'athlètes à une des 3 courses cyclistes dans l'ordre suivant: Date de la course au printemps (DCP), date de la course à l'été (DCE) et date de la course à l'automne (DCA). Si aucune date n'est présente on veut placer le 31 décembre de la même année spécifiquement.

Table "Courses":

	ATHLETE	DCP	DCE	DCA
1	MHP	.	2018-07-25	.
2	DV	2018-03-12	2018-07-25	2018-10-16
3	DP	.	.	.

MHP: Marie-Hélène Prémont

DV: David Veilleux

DP: Dominique Perras

COALESCE

Approche traditionnelle:

```
if      not missing(dcp) then dpartr=dcp;
else if not missing(dce) then dpartr=dce;
else if not missing(dca) then dpartr=dca;
else                          dpartr=mdy(12,31,2018);
```

Approche avec fonction:

```
dpartr=coalesce(dcp,dce,dca,mdy(12,31,2018));
```

Résultats: (DPARTR = Date de participation la plus récente)

	ATHLETE	DCP	DCE	DCA	DPARTR
1	MHP	.	2018-07-25	.	2018-07-25
2	DV	2018-03-12	2018-07-25	2018-10-16	2018-03-12
3	DP	.	.	.	2018-12-31

21

L'approche traditionnelle est plus longue à écrire que d'utiliser la fonction COALESCE.

Il existe également une fonction COALESCEC pour le traitement de chaînes de caractères.

SUBSTRN, FIRST et CHAR

```
retain ch "Québec  ";
s1=substrn(ch,1,1);
f1=first(ch);
c1=char(ch,1);
s2=substrn(ch,(lengthn(ch)),1);
f2=first(reverse(trim(ch)));
c2=char(ch,lengthn(ch));
```

	CH	S1	F1	C1	S2	F2	C2
1	Québec	Q	Q	Q	c	c	c

Longueurs respectives des variables résultantes:

	LCH	LS1	LF1	LC1	LS2	LF2	LC2
1	8	8	1	1	8	1	1

22

3 façons d'aller chercher le premier caractère d'une chaîne avec s1, f1, c1 et
3 façons d'aller chercher le dernier caractère d'une chaîne avec s2, f2, c2

Par rapport à SUBSTR, SUBSTRN permet des chaines vides et peut retourner une valeur de zéro pour une valeur manquante (caractère)

Pas vraiment intéressant d'utiliser la fonction CHAR avec 1 comme argument

Fonction LENGTHN: retourne la longueur d'une chaîne de caractères sans tenir compte des espaces à la fin.

|| vs PUT

```
datpay=today();
IF   DAY(DATPAY) < 10 THEN JOUR = '0'||  DAY(DATPAY);
IF   DAY(DATPAY) >  9 THEN JOUR =        DAY(DATPAY);
IF MONTH(DATPAY) < 10 THEN MOIS = '0'||MONTH(DATPAY);
IF MONTH(DATPAY) >  9 THEN MOIS =      MONTH(DATPAY);
pjour = put(  day(datpay),z2.);
pmois = put(month(datpay),z2.);
```

	DATPAY	JOUR	MOIS	PJOUR	PMOIS
1	2019-05-07	0 7	0 5	07	05

NOTE: Numeric values have been converted to character values at the places given by: (Line):(Column). 33:42 34:42 35:42 36:42

Nom	Type	Longueur
DATPAY	Date	8
JOUR	Alphanumérique	13
MOIS	Alphanumérique	13
PJOUR	Alphanumérique	2
PMOIS	Alphanumérique	2

23

Exemple trouvé dans un traitement réel.

En majuscule on voit le code original et en minuscule on voit le code que j'ai utilisé.

La fonction PUT en dépit de son existence lointaine, ne semble pas être suffisamment connue comme moyen d'effectuer la transformation de valeurs de variables.

Combinée avec le format "Z" on obtient assez facilement ce qu'on veut: le jour et le mois sur 2 positions incluant les cas ou on a besoin d'un zéro "non significatif"

CATS vs PUT

```
datpay=today();
IF   DAY(DATPAY) < 10 THEN JOUR = cats('0',  DAY(DATPAY));
IF   DAY(DATPAY) >  9 THEN JOUR = cats(      DAY(DATPAY));
IF MONTH(DATPAY) < 10 THEN MOIS = cats('0',MONTH(DATPAY));
IF MONTH(DATPAY) >  9 THEN MOIS = cats(    MONTH(DATPAY));
pjour = put(  day(datpay),z2.);
pmois = put(month(datpay),z2.);
```

	DATPAY	JOUR	MOIS	PJOUR	PMOIS
1	2019-05-07	07	05	07	05

~~NOTE: Numeric values have been converted to character values at the places given by: (Line):(Column). 33:42 34:42 35:42 36:42~~

Nom	Type	Longueur
DATPAY	Date	8
JOUR	Alphanumérique	200
MOIS	Alphanumérique	200
PJOUR	Alphanumérique	2
PMOIS	Alphanumérique	2

24

Avec la fonction CATS on obtient de bons résultats sans NOTE dans le journal
Cependant la longueur des chaînes résultantes avec CATS est de 200.

Il est donc recommandé de définir explicitement la longueur de variables caractères

Donc, la fonction PUT demeure le meilleur choix.

Informations sur les formats

On peut retrouver de l'information sur les formats dans le dictionnaire interne de SAS (DICTIONARY.FORMATS ou SASHELP.VFORMAT) ou grâce à la fonction **FMTINFO**.

```
retain f "nlmny";
c=fmtinfo(f,"cat");
t=fmtinfo(f,"type");
w=fmtinfo(f,"defw");
d=fmtinfo(f,"desc");
```

	F	C	T	W
1	nlmny	curr	BOTH	9

D
Ecrit le format monétaire de l'expression locale selon les paramètres régionaux spécifiés

25

Comment obtenir de l'information sur un format: on peut utiliser la fonction FMTINFO ou le dictionnaire.
Plusieurs types d'information à obtenir:
CAT, TYPE, DESC, MIND, MAXD, DEFD, MINW, MAXW, DEFW

Par contre dans le dictionnaire on ne retouve pas de champs DESC (Description)

La description en français semble venir de la version de Windows et non pas de l'option LOCALE

La documentation ne précise pas tout à fait les valeurs disponibles pour l'option CAT.

Le type retourné par la fonction FMTINFO ne correspond pas à ce
qu'on retrouve dans le champs FMTTYPE du dictionnaire

Les formats disponibles sont ceux de SAS et ceux créés par l'utilisateur

Dictionnaire des formats

Variables disponibles dans le dictionnaire:

```
create table DICTIONARY.FORMATS
  (
   libname char(8) label='Library Name',
   memname char(32) label='Member Name',
   path char(1024) label='Pathname',
   objname char(32) label='Object Name',
   fmtname char(32) label='Format Name',
   fmttype char(1) label='Format Type',
   source char(1) label='Format Source',
   minw num label='Minimum Width',
   mind num label='Minimum Decimal Width',
   maxw num label='Maximum Width',
   maxd num label='Maximum Decimal Width',
   defw num label='Default Width',
   defd num label='Default Decimal Width'
  );
```

Pour obtenir la description d'une table ou d'une vue du dictionnaire SAS
utiliser l'énoncé DESCRIBE de la procédure SQL;

Exemple:

```
proc sql;
  describe table dictionary.formats;
  describe view sashelp.vformat;
quit;
```

Rappel: code vs fonction

```
IF SUBSTR(CODE,17,02)='.0'
   THEN CODE = SUBSTR(CODE,1,17)||SUBSTR(CODE,19,02);
IF SUBSTR(CODE,17,02)='.0'
   THEN CODE = SUBSTR(CODE,1,17)||SUBSTR(CODE,19,02);
IF SUBSTR(CODE,16,02)='.0'
   THEN CODE = SUBSTR(CODE,1,16)||SUBSTR(CODE,18,03);
IF SUBSTR(CODE,16,02)='.0'
   THEN CODE = SUBSTR(CODE,1,16)||SUBSTR(CODE,18,03);
IF SUBSTR(CODE,15,02)='.0'
   THEN CODE = SUBSTR(CODE,1,15)||SUBSTR(CODE,17,04);
IF SUBSTR(CODE,15,02)='.0'
   THEN CODE = SUBSTR(CODE,1,15)||SUBSTR(CODE,17,04);
IF SUBSTR(CODE,14,02)='.0'
   THEN CODE = SUBSTR(CODE,1,14)||SUBSTR(CODE,16,05);
IF SUBSTR(CODE,14,02)='.0'
   THEN CODE = SUBSTR(CODE,1,14)||SUBSTR(CODE,16,05);
IF SUBSTR(CODE,13,02)='.0'
   THEN CODE = SUBSTR(CODE,1,13)||SUBSTR(CODE,15,06);
IF SUBSTR(CODE,13,02)='.0'
   THEN CODE = SUBSTR(CODE,1,13)||SUBSTR(CODE,15,06);
IF SUBSTR(CODE,12,02)='.0'
   THEN CODE = SUBSTR(CODE,1,12)||SUBSTR(CODE,14,07);
IF SUBSTR(CODE,12,02)='.0'
   THEN CODE = SUBSTR(CODE,1,12)||SUBSTR(CODE,14,07);
IF SUBSTR(CODE,11,02)='.0'
   THEN CODE = SUBSTR(CODE,1,11)||SUBSTR(CODE,13,08);
IF SUBSTR(CODE,11,02)='.0'
   THEN CODE = SUBSTR(CODE,1,11)||SUBSTR(CODE,13,08);
```

etc, etc.

60 lignes de code au total!!!

```
CodeTran=transtrn(CodeTran,".00",".");
CodeTran=transtrn(CodeTran,".0" ,".");
```

Liste ces codes originaux et transformés

Obs.	CodeLu	CodeTran
1	1.02.002.002.001.001	1.2.2.2.1.1
2	2.04.005.06.107.002	2.4.5.6.107.2
3	3.04.050.020.009	3.4.50.20.9
4	4.03.030.014	4.3.30.14

27

Dans une présentation antérieure « Bon code, bad code » Qu’est-ce du bon code SAS? on voulait effectuer une opération de suppression de zéros.

Dans l’approche originale, on avait besoin de 60 lignes de code pour y arriver.

Une approche avec fonction a réduit le code à deux lignes…

Vive les fonctions! Pour réduire le code substantiellement dans cet exemple.

Syntaxe: TRANSTRN(source, cible, remplacement)

Sagesse ancestrale

Thus spake the master programmer:

« Though a program be but three lines long, someday it will have to be maintained. »

© The Tao of Programming, Geoffrey James

Conclusion

Proposition: "Codons moins, codons mieux"

Les fonctions sont très nombreuses mais constituent un bon moyen de produire moins de code mais du code robuste et fiable (mais faire quelques essais cependant)

Alors, oui les fonctions et routines peuvent nous être utiles et sont en fait indispensables et incontournables de toutes façons

Allez-y! Faites de la magie avec les fonctions et partagez vos découvertes.

Comme Severus Rogue (Harry Potter)
était maître des potions
Devenez maître des fonctions!

29

Je lance un appel à toutes et à tous.

Partagez vos trouvailles, vos expériences dans l'utilisation des routines et fonctions.

On pourrait en faire un résumé lors de notre prochain rendez-vous.

Severus Rogue, maitre des potions. Vous pouvez devenir maître des fonctions!

Questions ?
Commentaires ?
Suggestions ?

Merci!

Contact:

```
1 data _null_;
2   courriel=lowcase(cat(reverse("Alain"),reverse("Voyer"),"@gmail.com"));
3   putlog "NOTE: " courriel=;
4 run;

NOTE: courriel=nialareyov@gmail.com
```

Marques de commerce

SAS et tous les autres noms de produits et services de SAS Institute Inc. sont des marques de commerce de SAS Institute Inc. aux États-Unis et dans les autres pays. ® indique l'inscription au États-Unis. Les autres noms de produits ou marques sont des marques déposées ou des marques de commerce de leurs compagnies respectives.

30

Merci!

Alain Voyer
Analyste en informatique (infrastructures technologiques)
Formateur pigiste en SAS
Mentor pour la plateforme z/OS