

PUT og INPUT funktionerne

Af:

Peter Kellberg
Danmarks Statistik
Sejrøgade 11
DK-2100 København Ø
pke@dst.dk

PUT og INPUT-funktionerne

Denne artikel er foranlediget af en henvendelse til vores interne SAS® support her i Danmarks Statistik. Hvad er egentlig den store forskel på PUT og INPUT-funktionerne. Tja...! Hvornår skal man vælge den ene frem for den anden? Tjo...! Og hvad med PUTC, PUTN, INPUTN og INPUTC? Få svarene her!

1 Introduktion

Allerførst: PUT- og INPUT-funktionerne er DATAstep-funktioner. Ved du for øvrigt, at der er 474 DATAstep funktioner i SAS®9? Ganske imponerende!

De fleste kender PUT- og INPUT-sætningerne, og en vis analogi er da også til at få øje på. Med INPUT-sætningen er det muligt at læse noget ind via et informat, og på den måde at få transformeret det indlæste til noget andet fx

```
input ... @123 regdato ddmmyy10. ...;
```

I dette tilfælde indlæses en dato fx 10/12/2005, som så forvandles til en SAS® dato, altså et antal dage efter 1. januar 1960.

Med PUT-sætningen er det muligt at skrive noget ud til en fil via et format, og på den måde at få transformeret det skrevne

```
put ... @123 branche $brfmt. ...;
```

I dette tilfælde har vi en branchekode i hånden, som så forvandles til en måske mere sigende tekst, når oplysningen skrives ud.

Det samme gør sig gældende for PUT- og INPUT-funktionerne: PUT afleverer noget via et **format**, mens INPUT afleverer noget via et **informat**. Så logikken i virkemåden er altså langt hen ad vejen den samme. Og dermed forvirringen.

Men der er lige den spidsfindighed, at PUT ALTID returnerer noget tekst (af typen CHAR).

Lad os se på en konkret problemstilling:

Vi har en tekstvariabel DATO indeholdende en dato i klar tekst fx 10/12/2006. Vi vil nu oprette en ny variabel, som skal indeholde den tilsvarende SAS dato. Hvad gør vi? Vi kunne jo starte med at spørge:

- Findes der et **format** som laver 10/12/2006 om til en SAS dato?
 - Nej! Dette udelukker brugen af PUT, som bruger formater.
- Findes der et **informat**, som kan omsætte 10/12/2006 til en SAS dato?
 - Ja, det er informatet ddmmyy10.! Så bruger vi INPUT, som jo bruger informater.
 - SASdato=input(dato,ddmmyy10.);

Lige præcis i dette tilfælde er det tilstedeværelsen af det rigtige informat, der aftvinger brugen af INPUT.

I oversigtsform:

Funktion	Benytter	Afleverer
INPUT	Informater	Noget numerisk eller tekst afhængigt af omstændighederne
PUT	Formater	Altid noget tekst

2 PUT, Typisk anvendelse

1) Når man vil konvertere noget **numerisk til tekst**.

```
nyvar=put(numvar,4.);
```

Fordelen her er, at det er os, der bestemmer, hvilket format, der skal anvendes. Hvis vi fx ville have foranstillede nuller, kunne vi jo bare bruge:

```
nyvar=put(numvar,z4.);
```

i stedet.

Variablen på venstre side er typisk en tekstvariabel.

2) Hvis man vil have den formaterede værdi af en variabel i en selvstændig variabel fx:

```
koen_txt=put(koen_kode, koenfmt.);  
dato_txt=put(dato, ddmmyy10.);
```

3 INPUT, Typisk anvendelse

1) Når man vil konvertere noget **tekst til numerisk**.

```
nyvar=input(tekstvar,4.);
```

eller hvis du fx (af skæbnens ugunst) har en datovariabel i klar tekst, som du vil lave til en SAS[®] dato:

```
tekst_dato='12/01/2003';  
sasdato=input(tekst_dato,ddmmyy10.);
```

4 Automatiske konverteringer

Med PUT- og INPUT-funktionerne er det altså muligt at få lavet noget om fra NUM til CHAR og omvendt. Og vi kan endda via vores formater og informatier samtidig få transformeret indholdet til noget andet. Hvis det bare handler om simpel konvertering fra NUM til CHAR eller omvendt, så kan SAS[®] langt hen ad vejen gøre det for os, uden at vi skal spekulere meget over det.

Automatiske konverteringer laver SAS, når der ikke "er orden i tingene". Tingene skal passe sammen! Fx vil SAS ikke have, at der indgår tekst i beregninger. Sådanne automatiske konverteringer vil altid udløse meddelelser i loggen a la:

```
NOTE: Character values have been converted to numeric values at the places given  
by: (Line):(Column).  
4:9
```

Lad os se på et eksempel, hvor vi har et årstal i en tekstvariabel AAR, og nu skal vi have lagt ét år til:

```
data test;
  aar='2003';
  slutaar=aar+1;
run;
```

SAS® bryder sig ikke om, at tekst indgår i beregninger. Det vil altid blive konverteret. Så i dette tilfælde vil teksten "2003" blive konverteret til tallet 2003 inden der lægges 1 til. Dette vil naturligvis fremgå af loggen.

Generelt kan man opstille følgende regler for, hvornår der vil finde konvertering sted:

- Hvis en tekstvariabel bruges i sammenhæng med en numerisk operator, vil indholdet blive konverteret til numerisk. Det har vi lige set et eksempel på!
- Hvis vi sammenligner noget tekst med noget numerisk fx i en IF-sætning, vil teksten blive konverteret til numerisk, før sammenligningen finder sted.
- Hvis en numerisk variabel bruges i sammenhæng med en tekstoperator, vil indholdet blive konverteret til tekst. SAS bruger altid BEST12. formatet til det, og resultatet af konverteringen HØJREstilles. Fx:

```
data test;
  aar=2003;
  txt='Opgørelse pr.'!!aar;
run;
proc print;
run;
```

Dette giver følgende udskrift:

Obs	aar	txt	
1	2003	Opgørelse pr.	2003

Dette problem kan løses på forskellig måde, men med én af de nye konkateneringsfunktioner går det nemt:

```
data test;
  aar=2003;
  txt=catx(' ', 'Opgørelse pr.', aar);
run;
proc print;
run;
```

- Hvis man har en numerisk variabel på venstre side af en tildelingssætning samt en tekstvariabel på højre side, vil indholdet af tekstvariablen blive konverteret til numerisk.
- Hvis man har en tekstvariabel på venstre side af en tildelingssætning samt en numerisk variabel på højre side, vil indholdet af den numeriske variabel blive konverteret til tekst. I dette tilfælde bruges BESTn. formatet, hvor "n" vil være længden på den modtagende tekstvariabel.

Så i mange situationer kan man lade SAS om at stå for konverteringen. Det er der sådan set ikke noget odiøst i, så længe dette ikke giver anledning til fejl, som man vil kunne tackle med PUT/INPUT-funktionerne.

Et eksempel:

Vi har et CPR-nummer i en numerisk variabel CPRNR, og vi vil nu gerne have dette over i en tekstvariabel CPRNUM med længden 10. Så hvad er mere nærliggende end:

```
cprnum=cprnr;
```

Men hvad med det foranstillede nul i fx 0205791234? Det kommer jo ikke med! Så i dette tilfælde er der ingen vej uden om PUT-funktionen:

```
cprnum=put(cprnr,z10.);
```

Mange bryder sig ikke om meddelelser om automatisk konvertering, og på én måde kan det være forståeligt nok: Hvis der er et eller andet galt, er det jo ikke underligt, hvis man som en del af problemløsningsadfærden retter opmærksomheden på netop de automatiske konverteringer. Men langt hen ad vejen gør SAS® tingene OK, man kan så evt. skrive en lille kommentar i koden om, hvorfor der her sker en konvertering, og at den er uproblematisk. Med hensyn til performance er der ikke nogen forskel på at gøre tingene selv eller lade SAS® systemet om det.

5 Når INPUT får noget galt i halsen

Hvis INPUT-funktionen får noget galt i halsen, vil der komme en meddelelse i loggen om det.

Et eksempel: Vi har fået leveret nogle lønoplysninger. Lønnen findes i en tekstvariabel INDK_TXT. Nogle af lønoplysningerne er klassificerede og indeholder teksten "class". Vi vil naturligvis gerne have en numerisk variabel med lønnen, så vi kan begynde at regne på den. Så hvad med fx:

```
indkomst=input(indk_txt,5.);
```

Dette vil uvægerligt medføre en note i loggen, når INPUT-funktionen støder på teksten "class":

```
239 data indkomster;
240     set test;
241     indkomst=input(indk_txt, 5.);
242 run;
```

```
NOTE: Invalid argument to function INPUT at line 241 column 12.
indk_txt=class indkomst=. _ERROR_=1 _N_=3
NOTE: Mathematical operations could not be performed at the following places.
The results of the operations have been
      set to missing values.
      Each place is given by: (Number of times) at (Line):(Column).
      1 at 241:12
```

Disse meddelelser kan elimineres med ? eller ?? på baggrund af din velovervejede beslutning (og kommentar) om at gøre det.

Argument	Virkning
?	Fjerner meddelelsen. _ERROR_ sættes til 1, og den pågældende observation listes
??	Fjerner meddelelsen. _ERROR_ er urørt, og den pågældende observation listes ikke. Man kan med andre ord ikke se NOGET mistænkeligt i loggen.

fx

```
indkomst=input(indk_txt,?? 5.);
```

TIP

? og ?? kan også benyttes ifm INPUT-sætningen.

6 PUTC, PUTN, INPUTC og INPUTN.

PUT- og INPUT-funktionerne forventer, at man skriver navnet på et format eller informat. Så virkemåden ligger altså klar allerede på kompileringstidspunktet. Med de øvrige funktioner har vi mulighed for at lade formatnavnet udvikle sig på eksekveringstidspunktet.

Funktion	Formatangivelsen skal udvikle sig til
PUTC	Et tekstformat
PUTN	Et numerisk format
INPUTC	Et tekst informat
INPUTN	Et numerisk informat

Lad os tage et eksempel:

Vi har registreret en begivenhed, som enten har fundet sted på kommune- eller amtsniveau. For den aktuelle begivenhed er således registreret enten en kommune- eller amtskode. Vi vil nu gerne se i klar tekst, hvor denne begivenhed er registreret. Til formålet laver vi 2 overskuelige formater til hhv. kommuner og amter:

```
proc format;
  value $amt '12'='Københavns amtskommune'
            other='Andet amt';
  value $kom '101'='Københavns kommune'
            other='Anden kommune';
run;
```

og vi skal lige have nogle testdata:

```
data test;
  input kode $ 1-3;
cards;
12
101
;
run;
```

Problemet er, at vi skal benytte forskellige formater alt efter hvor mange cifre, der er i koden.

Der er intet i vejen for, at vi kan kode os ud af det på almindelig vis:

```
data test2;
  set test;
  if length(kode)=2 then
    tekst=put(kode, $amt.);
  else
    tekst=put(kode, $kom.);
run;
```

En udmærket løsning – nemt overskuelig.

En variant af denne kunne være:

```
data test2;
  length tekst $ 25;
  set test;
  if length(kode)=2 then
    formatnavn='$amt';
  else
    formatnavn='$kom';

  tekst=putc(kode, formatnavn);
run;
```

Her har vi gjort brug af den lidt mere dynamiske variant af PUT-funktionen, PUTC. Men kommer der flere varianter af koder til fx en 4 cifret kode, ja så skal vi rette i DATAstep-koden. Så det er stadigvæk ikke helt dynamisk.

Det kunne være super, hvis vi kunne lave et format, som kunne styre hvilket format, der skal benyttes alt efter hvilken længde, koden har. Så lad os prøve det:

```
proc format;
  value lgdfmt 2='$amt'
              3='$kom';
  value $amt  '12'='Københavns amtskommune'
            other='Andet amt';
  value $kom  '101'='Københavns kommune'
            other='Anden kommune';
run;
```

Nu har vi tilføjet et numerisk format LGDFMT, som omsætter længden til navnet på et format.

Så ser det således ud:

```
data test2;
  length tekst $25;
  set test;
  tekst=putc(kode, put(length(kode),lgdfmt.));
run;
```

Kommer der nu flere regionale inddelinger til, skal der blot ændres i formatet LGDFMT og der skal så tilføjes formater svarende til de nye inddelinger. SAS® koden skal der ikke pilles i!

Ovenstående kunne måske være skrevet i lidt flere tempi for overskuelighedens skyld:

```
data test2;
  length tekst $25;
  set test;
  laengde=length(kode);
  formatnavn=put(laengde,lgdfmt.);
  tekst=putc(kode, formatnavn);
run;
```

7 PUTC, PUTN, INPUTC og INPUTN i makromiljøet.

De fleste DATAstep-funktioner kan kaldes i makromiljøet med %SYSFUNC og %QSYSFUNC. Dette gælder tillige PUTC, PUTN, INPUTC og INPUTN, men IKKE PUT og INPUT.

Formatted: Danish

På den måde er det fx en smal sag at få transformeret en makrovariabel fra kode til tekst:

```
proc format;
  value koefmt 1='Mand'
              2='Kvinde'
              other='Køn ikke oplyst';
run;
%let koenkode=1;
%let koentekst=%sysfunc(putn(&koenkode,koefmt.));
%put _user_;
```

I loggen ses

```
GLOBAL KOENKODE 1
GLOBAL KOENTEKST Mand
```