Stat 302
Statistical Software and Its Applications
SAS Functions

Yen-Chi Chen

Department of Statistics, University of Washington

Autumn 2016

Creating New Variables

- Here we create new variables using functions applied to existing variables.
- We will do this by example, showing code and output.
- This is based on Ch. 11 in Cody's book.

A List of Functions

More Math	
Operator	Description
ABS(x)	Absolute value of x $= \|x\|$
EXP(x)	Exponential of x $= e^x$
INT(x)	Truncate x to an integer
LOG(x)	Natural log of x, $\ln(x) = \log_e(x)$
LOG10(x)	Log base 10 of x $= \log_{10}(x)$
MOD(x,d)	Remainder when x is divided by d
ROUND(num)	Round num to the nearest integer
ROUND(num,unit)	Round num to the nearest specified unit
SQRT(x)	Square root of x

Example Code for Function Evaluations

```
data functions;
 input x;
 rndx = round(x); intx = int(x);
 sqx = x**2; sqrtx = sqrt(x);
 log10x = log10(x); lnx = log(x);
 absx = abs(x); expx = exp(x);
 datalines;
 4
 9.3
 -9.3
 run;
title "Function Evaluations";
proc print data=functions noobs; run;
```

Output for Function Evaluations

x	rndx	intx	sqx	sqrtx	log10x	lnx	absx	expx
4.0	4	4	16.00	2.00000	0.60206	1.38629	4.0	54.60
9.3	9	9	86.49	3.04959	0.96848	2.23001	9.3	10938.02
-9.3	-9	-9	86.49	.	.	.	9.3	0.00

- We need to use `x**2` to compute x^2.

Using a Do Loop – 1

```
data equation;
   do X = 0 to 10 by 1;
        Y = 2*x+8;
        output;
    end;
run;
title " X Y values";
proc
print data = equation noobs;
run;
```

Using a Do Loop – 2

X Y values

X	Y
0	8
1	10
2	12
3	14
4	16
5	18
6	20
7	22
8	24
9	26
10	28

Using a Do Loop – 3: The `output;` Statement

- The `output;` statement instructs SAS to write out an observation to the output data set.
- An output usually occurs at the bottom of the data step.
- When you include an `output;` statement anywhere within the data step, SAS does not execute an automatic output at the bottom of the data step.

Using a Do Loop – 4

```
do x = 1,2,5,10;

( values of x are: 1, 2, 5, 10 )

do month = 'Jan' 'Feb' 'Mar';

( values of month are: 'Jan', 'Feb', 'Mar' )

do n = 1,3, 5 to 9 by 2, 100 to 200 by 50

( values of n are: 1, 3, 5, 7, 9, 100, 150, 200 )
```

Effect of the Sum Statement Solution – 1

```
data compound;
  Interest = .0125;
  Total = 100;
  do Year = 1 to 5;
    Total + Total * Interest;
    output;
  end;
run;
title "Listing of Compound";
proc print data=compound noobs;
run;
```

Effect of the Sum Statement Solution – 2

Listing of Compound

Interest	Total	Year
0.0125	101.250	1
0.0125	102.516	2
0.0125	103.797	3
0.0125	105.095	4
0.0125	106.408	5

Effect of the Sum Statement Solution – 3

Form of the Sum Statement:

`variable+increment`

- `variable` is retained from data step to data step
- `variable` not automatically initialized as `.` (missing)
- `variable` is initialized at 0 on first data step
- Data steps with missing value in `increment` are ignored

Effect of the Sum Statement Solution – 4

```
data compound;
  Interest = .0125;
  Total = 100;
  do Year = 1 to 5;
    W = sqrt((-1)**Year);
    Total + Total * Interest*W;
    output;
  end;
run;
title "Listing of Compound";
proc print data=compound noobs;
run;
```

Effect of the Sum Statement Solution – 5

Listing of Compound

Interest	Total	Year	W
0.0125	100.000	1	.
0.0125	101.250	2	1
0.0125	101.250	3	.
0.0125	102.516	4	1
0.0125	102.516	5	.

IF statement – 1

```
data testif;
 input x y;
 z1 = x+2*y;
 z2 = x**2 -y;
 datalines;
 4 5
 3 1
 -2 7
 -10 -2
 run;
title "new data";
proc print data=testif noobs; run;
```

IF statement – 2

new data

x	y	z1	z2
4	5	14	11
3	1	5	8
-2	7	12	-3
-10	-2	-14	102

IF statement – 3

```
data testif;
 input x y;
 z1 = x+2*y;
 z2 = x**2 -y;
 if y > 5 then Group = 1;
 if z1 < 0  then Group = 2;
 if x > 0 and z1 >0 then Group =3;
 datalines;
 4 5
 3 1
 -2 7
 -10 -2
 run;
title "new data 2";
proc print data=testif noobs; run;
```

IF statement – 4

new data 2

x	y	z1	z2	Group
4	5	14	11	3
3	1	5	8	3
-2	7	12	-3	1
-10	-2	-14	102	2

You can use `if` to select the data.

IF statement – 5

```
data testif;
 input x y;
 z1 = x+2*y;
 z2 = x**2 -y;
 if x > 0 and z1 >0;
 datalines;
 4 5
 3 1
 -2 7
 -10 -2
 run;
title "new data 2";
proc print data=testif noobs; run;
```

IF statement – 6

new data 2

x	y	z1	z2
4	5	14	11
3	1	5	8

`if x > 0 and z1 >0;` $\longrightarrow$ this outputs only those observations satisfy the conditions.

IF statement – 7

```
data testif;
 do X = 1 to 20 by 1;
  if (-1)**X >0 then output;
 end;
 run;
title "new data 3";
proc print data=testif noobs; run;
```

IF statement – 8

new data 3

X
2
4
6
8
10
12
14
16
18
20

`if (-1)**X >0 then output;` $\longrightarrow$ this argument means that whenever the if statement holds, it will output the value (from the loop).

IF statement – 9

```
data testif;
 do X = 1 to 20 by 1;
  if (-1)**X >0 then do;
  Y = 3+2*X;
  output;
  end;
 end;
 run;
title "new data 4";
proc print data=testif noobs; run;
```

IF statement – 10

new data 4

X	Y
2	7
4	11
6	15
8	19
10	23
12	27
14	31
16	35
18	39
20	43

`if (-1)**X >0 then do;` $\longrightarrow$ you can do multiple operations when the `if` statement holds.

Descriptive Statistics – 1

```
data psych;
        input ID $ Q1-Q10;
        if n(of Q1-Q10) ge 7 then Score = mean(of Q1-Q10);
        MaxScore = max(of Q1-Q10);
        MinScore = min(of Q1-Q10);
datalines;
001 4 1 3 9 1 2 3 5 . 3
002 3 5 4 2 . . . 2 4 .
003 9 8 7 6 5 4 3 2 1 5
;
title "Descriptive Stats";
proc print data = psych noobs;
* var Score maxScore MinScore;
run;
```

Descriptive Statistics – 2

Descriptive Stats

ID	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Score	MaxScore	MinScore
001	4	1	3	9	1	2	3	5	.	3	3.44444	9	1
002	3	5	4	2	.	.	.	2	4	.	.	5	2
003	9	8	7	6	5	4	3	2	1	5	5.00000	9	1

Descriptive Statistics – 3

Descriptive Stats

Score	MaxScore	MinScore
3.44444	9	1
.	5	2
5.00000	9	1

Descriptive Statistics – 4

- The **N** function returns the number of non-missing numeric values among its arguments.
- A companion function NMISS returns the number of missing numeric values among its arguments.
- In all such functions you must precede the list of variables Var1-Varn with the key word OF, otherwise SAS assumes that you subtract Varn from Var1.
- The MEAN function ignores missing values.
  This also applies to other such functions.

Order Statistics – 1

```
data order_stats;
        set psych;
        * assumes that psych was created previously
        and is still in the WORK library;
        M3 = mean(largest(1,of Q1-Q10),
                largest(2,of Q1-Q10),largest(3,of Q1-Q10),
                smallest(1,of Q1-Q10),smallest(2,of Q1-Q10),
                smallest(3,of Q1-Q10));
        * computes the average of the 3 extreme
                observations from each end;
        run;
title "Mean of Order Statistics";
proc print data = order_stats noobs;
var Q1-Q10 M3;
run;
```

Mean of Order Statistics

Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	M3
4	1	3	9	1	2	3	5	.	3	3.66667
3	5	4	2	.	.	.	2	4	.	3.33333
9	8	7	6	5	4	3	2	1	5	5.00000

In-class Exercise

- Creates a dataset with 3 variables `A, B, C, D`. Use a `do` loop for variable `A` that takes values `3,6,9,12,15,18,21`. Moreover, `B = A-1`, `C = (A-10)**2`, `D = C-A`. Print out the dataset.
- Add a new variable `E` such that if `C < 10`, `E = 0`; if `C` $\geq$ `10, E = 1`. Print out the dataset.
- Print out the dataset for those observations with `D < 0` or `C <= 6`.