Avoiding Macros in SAS

Fareeza Khurshed

Yiye Zeng

Macro's

- Usually used to avoid repetitive code or automate procedures
- I use them and write them on a regular basis
- Based on MY experience there's a set of common questions that are asked regarding how to write a macro

FAQ

- This presentation goes over those questions and how to answer them – without using macros
- Ends up being more tips and tricks or an FAQ document

Q: Export a SAS dataset to separate TXT files?

- Use the FILEVAR option in a file statement
- Sample Code on GitHub
- Documentation (SAS 9.4)

Q: Export a SAS dataset to separate TXT files?

- Export CARS dataset, with a unique file for each MAKE
- File name should be MAKE
- Each file should have the same column headers

```
PROC SORT DATA=SASHELP.CARS OUT=CARS; BY make; RUN;

DATA _NULL_;

SET cars; *Dataset to be exported;
BY make; *Variable that file is to be split on;

*Create path to file that is to be exported;
if first.make then out_file=cats('/folders/myfolders/', trim(make));

file temp filevar=out_file dlm=',' dsd;

*If first value of make then output column names;
if first.make then put 'Make, Model, MPG_HIGHWAY, MPG_CITY';

*Output variables;
put make model mpg_highway mpg_city;

run;
```

Dynamic file name!

Q: Import multiple txt files?

- Create list of files and use FILEVAR in infile
- Wildcards in file statement
- SAS code on GitHub

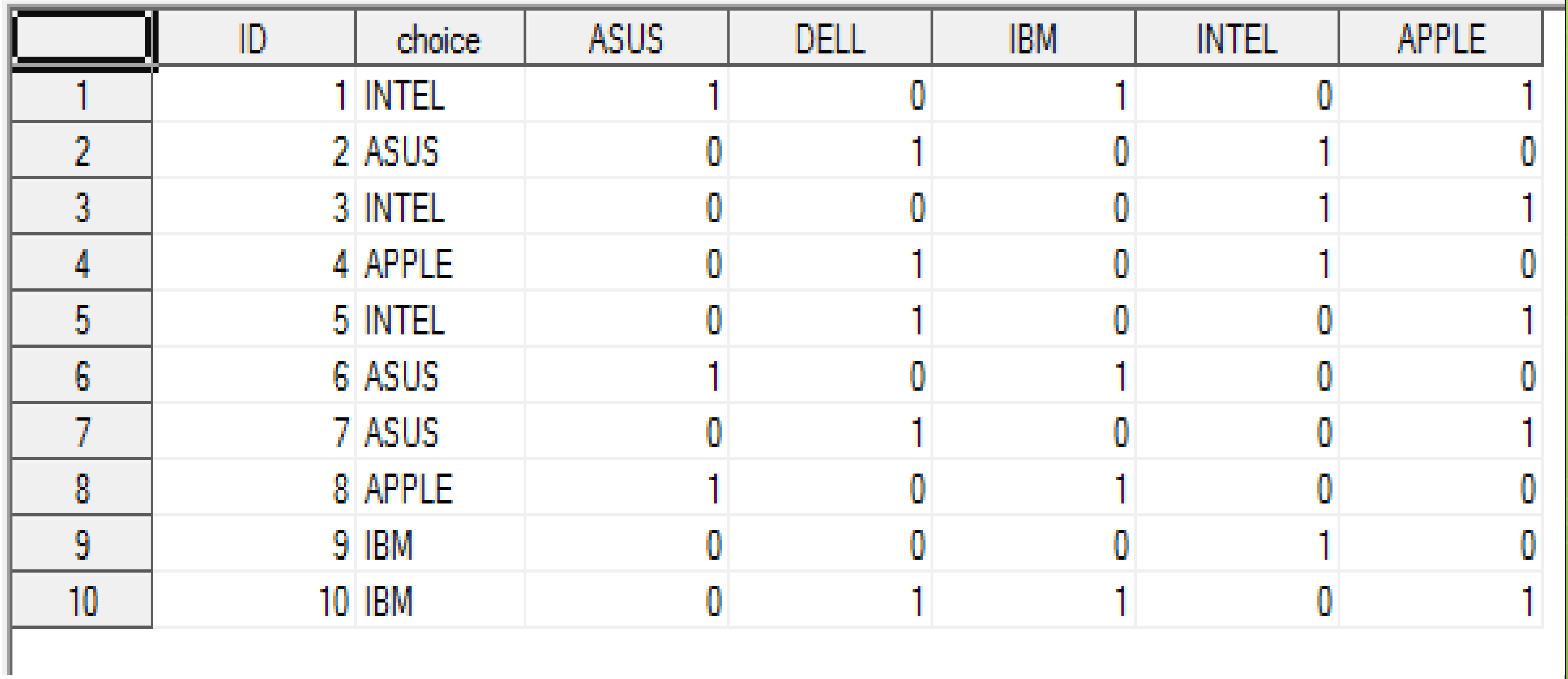
```
data import_all;

*make sure variables to store file name are long enough;
length filename txt_file_name $256;

*keep file name from record to record;
retain txt_file_name;

*Use wildcard in input;
infile "Path\*.txt" eov=eov filename=filename truncover;

*Input first record and hold line;
input@;

*Check if this is the first record or the first record in a new file;
*If it is, replace the filename with the new file name and move to next line;
if _n_ eq 1 or eov then do;
 txt_file_name = scan(filename, -1, "\");
 eov=0;
end;

*Otherwise  go to the import step and read the files;
else input

 *Place input code here;

;
run;
```

Filename of the imported file is captured as a variable

This assumes that each file has column headers and uses the EOV option to account for it.

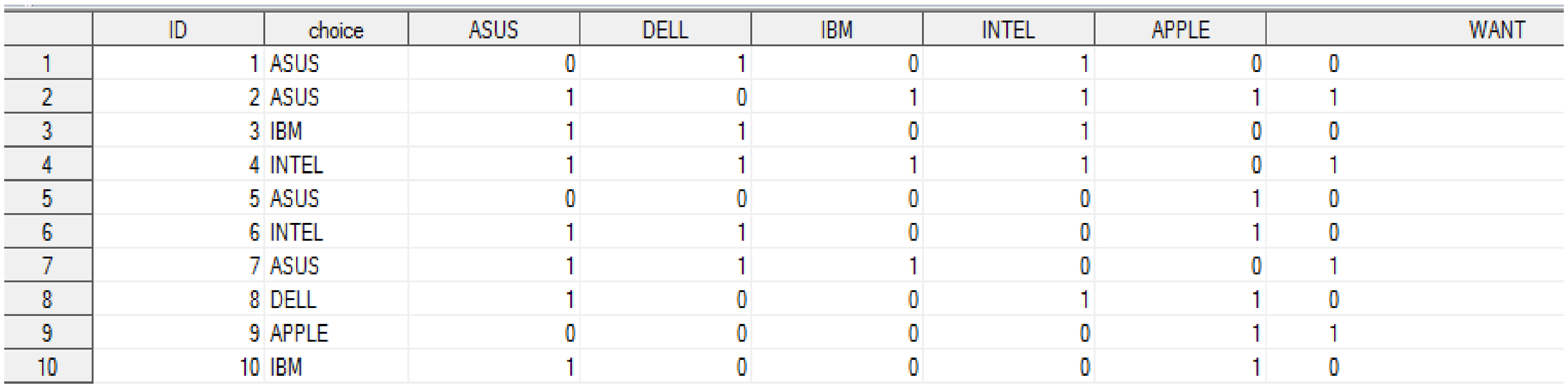
Q: Find the value of a specific variable?

- What is the value of the variable at choice?



	ID	choice	ASUS	DELL	IBM	INTEL	APPLE
1	1	INTEL	1	0	1	0	1
2	2	ASUS	0	1	0	1	0
3	3	INTEL	0	0	0	1	1
4	4	APPLE	0	1	0	1	0
5	5	INTEL	0	1	0	0	1
6	6	ASUS	1	0	1	0	0
7	7	ASUS	0	1	0	0	1
8	8	APPLE	1	0	1	0	0
9	9	IBM	0	0	0	1	0
10	10	IBM	0	1	1	0	1

WANT=vvaluex(choice);



	ID	choice	ASUS	DELL	IBM	INTEL	APPLE	WANT
1	1	ASUS	0	1	0	1	0	0
2	2	ASUS	1	0	1	1	1	1
3	3	IBM	1	1	0	1	0	0
4	4	INTEL	1	1	1	1	0	1
5	5	ASUS	0	0	0	0	1	0
6	6	INTEL	1	1	0	0	1	0
7	7	ASUS	1	1	1	0	0	1
8	8	DELL	1	0	0	1	1	0
9	9	APPLE	0	0	0	0	1	1
10	10	IBM	1	0	0	0	1	0

Q: Rename multiple variables?

- Use variable listing methods
  - Rename cc1-cc12=dd1-dd12;
- Use SASHELP(DICTIONARY) Tables to generate rename statement
- Map names from table – similar code can be used to apply labels or formats.

Old_Name	New_Name
Col1	ASUS
Col2	DELL
Col3	IBM
Col4	INTEL
Col5	APPLE
Col6	GOOGLE

```
proc sql noprint;
select catx("=", old_name, new_name)
   into :rename_list separated by " "
from rename_table;
quit;
%put &rename_list;
col1=ASUS col2=DELL col3=IBM col4=INTEL col5=APPLE
   col6=GOOGLE
proc datasets library=work nodetails nolist;
modify sample_data;
rename &rename_list;
run;quit;
```

Rename variables with common suffix?

- Rename a variable with a common suffix
- Easier with prefix – SAS offers shortcuts
- Example:
  - Rename all variables that end with _DATE with a prefix DT_

```
data sample;
do i=10000 to 12000;
        start_date=i;
        middle_date=i+3;
        end_date=i+5;
        date_no_change=start_date;
        output;
end;

format start_date end_date
middle_date date9.;
run;
```

```
proc sql noprint;
        select catx("=", name, catt('DT_',
        tranwrd(upper(name), '_DATE', ' ')))
        into :rename_list
        separated by " “

        from sashelp.vcolumn

        where libname='WORK'
        and memname='SAMPLE'
        and upper(trim(name)) like '%_DATE';
Quit;

%put &rename_list;
```

```
start_date=DT_START
middle_date=DT_MIDDLE
end_date=DT_END


proc datasets library=work nodetails nolist;
modify sample;
rename &rename_list;
run; quit;
```

```
                                                       date_no_
Obs    i      DT_START     DT_MIDDLE       DT_END      change

 1   10000   19MAY1987   22MAY1987   24MAY1987      10000
 2   10001   20MAY1987   23MAY1987   25MAY1987      10001
 3   10002   21MAY1987   24MAY1987   26MAY1987      10002
 4   10003   22MAY1987   25MAY1987   27MAY1987      10003
 5   10004   23MAY1987   26MAY1987   28MAY1987      10004
 6   10005   24MAY1987   27MAY1987   29MAY1987      10005
 7   10006   25MAY1987   28MAY1987   30MAY1987      10006
 8   10007   26MAY1987   29MAY1987   31MAY1987      10007
 9   10008   27MAY1987   30MAY1987   01JUN1987     10008
```

Q: Run multiple regressions?

- BY processing in general
- Multiple dependent variables in different columns?
- Transpose data first to create a BY variable!

Macros that are functions

- If you want a macro to return a value use PROC FCMP instead – mostly BASE SAS code – essentially creates a custom function

```
proc fcmp outlib=work.functions.conversions;

   function BMI(wgt_lb,ht_inches) ;
      BMI=(wgt_lb*703)/(ht_inches*ht_inches);
      return(BMI);
   endsub;

   Function lb2kg(lb);
      Kg=lb/2.2;
      return(kg);
   endsub;
run;
options cmplib=(work.functions);
data bmi;
   set sashelp.class(keep=name age weight height);
   BMI = bmi(weight,height);
   Weight_kg=lb2kg(weight);
run;
```

Q: Split a SAS dataset to multiple SAS datasets?

- Generally don't recommend it!!!
- Use BY group processing instead
- Or Call Execute

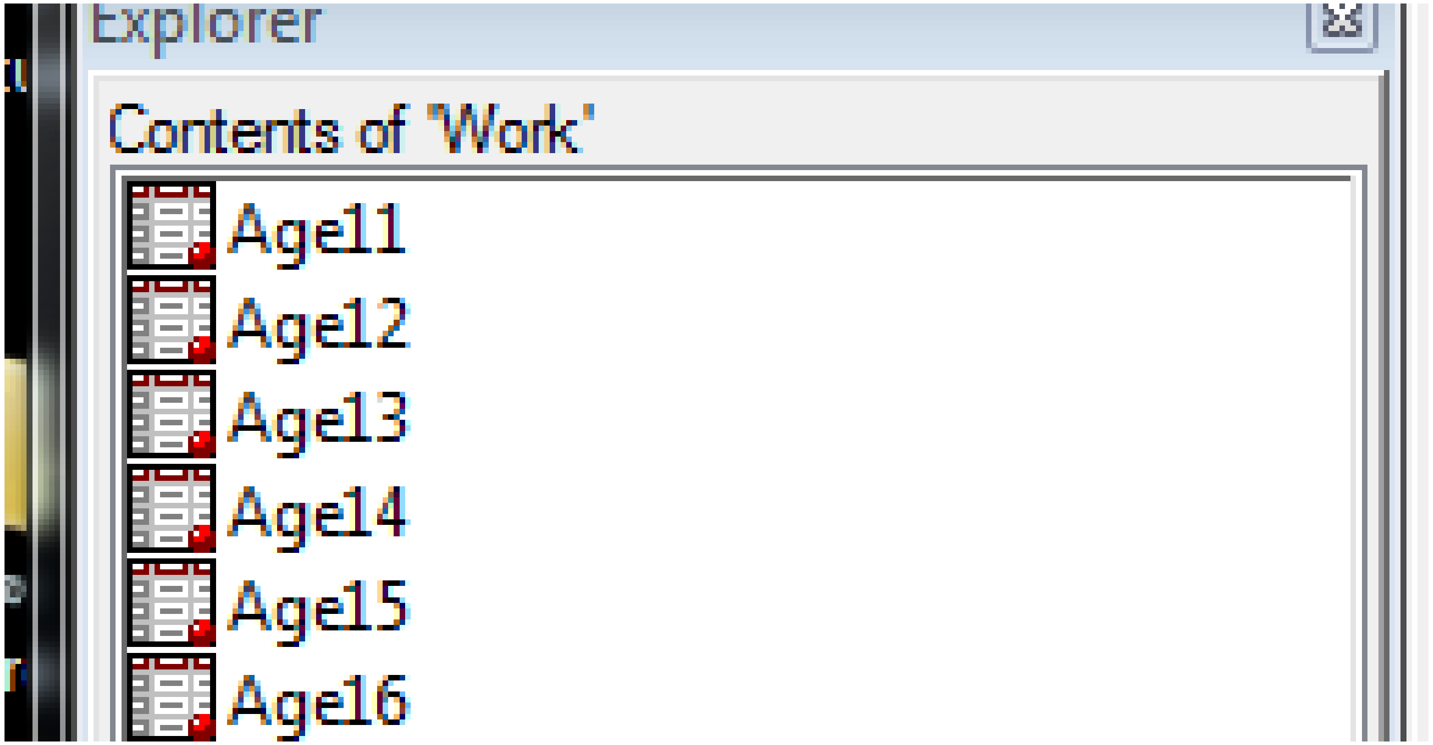
Another Call Execute Example

- Call Execute will run valid SAS code
- Generate the code in a string and use Call Execute to run the code
- Similar to macro but more easily data driven

Slipt data by Age

```
proc sort data=sashelp.class out=class; by age; run;

data code;
set class;
by age;

if first.age then do;
    string = cat('Data Age', age, '; set class; where age=',
age, ';run;');
    call execute(string);
    output;
end;

keep string;
run;
```

```
Obs              string

  1    Data Age11; set class; where age=11;run;
  2    Data Age12; set class; where age=12;run;
  3    Data Age13; set class; where age=13;run;
  4    Data Age14; set class; where age=14;run;
  5    Data Age15; set class; where age=15;run;
  6    Data Age16; set class; where age=16;run;
```


Explorer

Contents of 'Work'

Age11
Age12
Age13
Age14
Age15
Age16

Call a macro multiple times

- Call Execute
- Assume macro takes parameter which are stored in a dataset

Print data for all age-sex combinations

```
proc sort data=sashelp.class out=class;
by age sex;
run;

%macro summary(age=, sex=);

proc print data=sashelp.class;
   where age=&age and sex="&sex";
run;

%mend;
```

```
data sample;
set class;
by age sex;

if last.sex;
string =
    catt('%summary(age=', age, ',sex=', sex, ');');
put string;
run;
```

```
%summary(age=11,sex=F);
%summary(age=11,sex=M);
%summary(age=12,sex=F);
%summary(age=12,sex=M);
%summary(age=13,sex=F);
%summary(age=13,sex=M);
%summary(age=14,sex=F);
%summary(age=14,sex=M);
%summary(age=15,sex=F);
%summary(age=15,sex=M);
%summary(age=16,sex=M);
```

```
data _null_;
set sample;
call execute(string);
run;
```

***Executes macro after the data step!!**