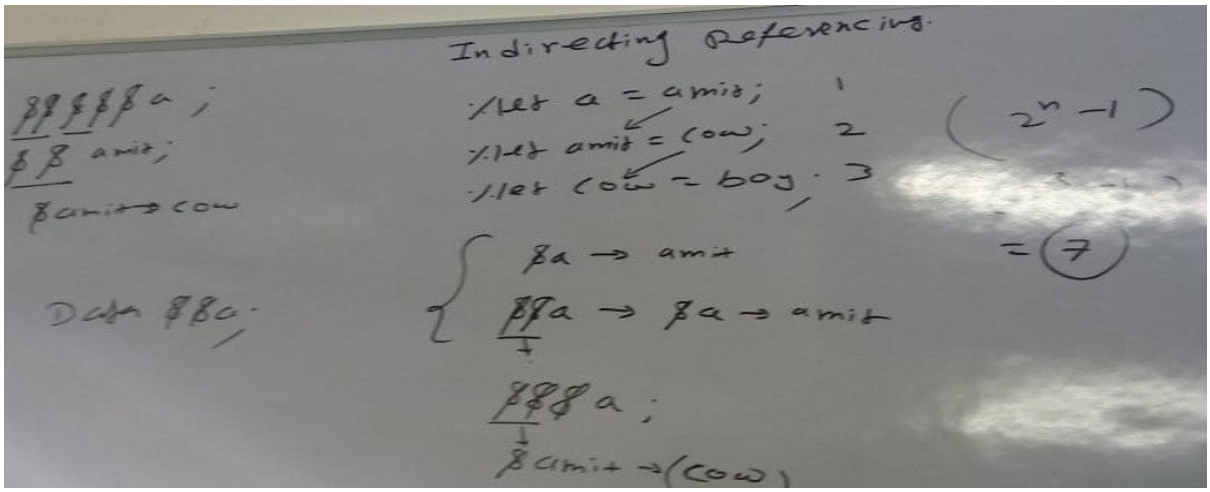
Statistical Analysis System: Class 23

Dated: 19/05/2018

Indirect referencing:

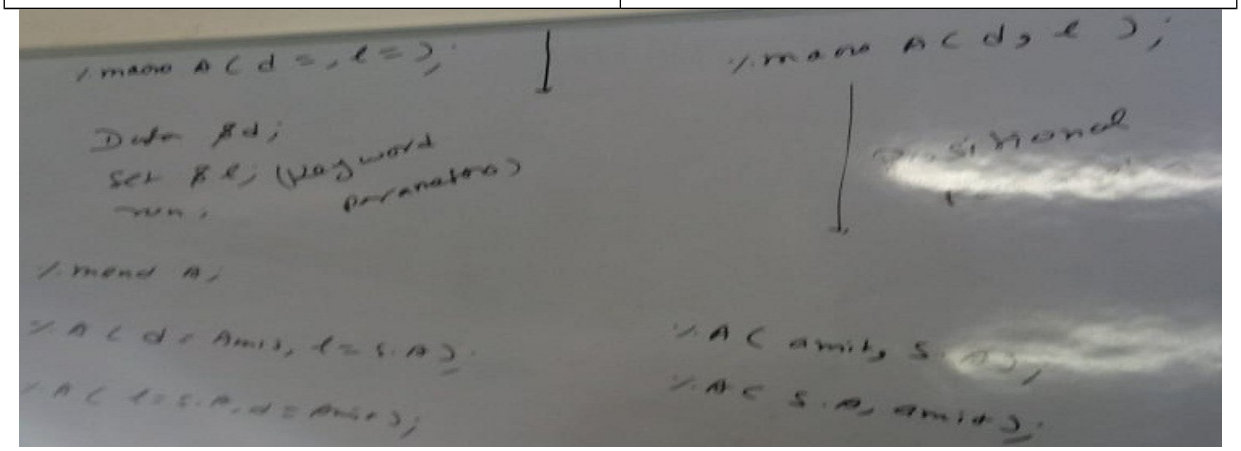
Using multiple ampersands (&) for reference to a macro variable is called indirect referencing.

- Ampersands (&), are resolved from left to right.
- Ampersands (&), are resolved in pair (groups of 2), i.e && &
- Number of ampersands required to resolve an indirect refrencing on a macro variable is 2^n-1



Keyword and Positional Parameter:

Keyword Parameter	Positional Parameter
• Defined as: macro a(d=, l=); • Keyword parameter comes 2nd while defining and also while passing values as parameter.	• Defined as: macro a(d, l); • Positional parameter comes before keyword parameter while defining and also while passing values as parameter.



4th way to create Macro variable - INTO:

INTO: variable_name separated by “delimiter” .Used with Proc SQL.

Prog 1

```
%macro exporty;

%let act=HIGH@LOW@MOD;
%do i=1 %to 3;

%let ac=%scan(&act,&i,"@");

data &ac;
set sasuser.admit;
where actlevel="&ac";
run;

Proc export data=&ac outfile= "C:\Documents and
Settings\sasadm\Desktop\2\admit.xls"
DBMS=excel replace;
Sheet="&ac";
run;

%end;
%mend exporty;

%exporty;
```

Explained: This macro generates admit.xls with 3 separate sheets "HIGH, LOW, MOD" as per the distinct values of variable actlevel .

Prog 2

```
%macro exporty;

proc sql noprint;
select distinct actlevel into: act separated by "@" from sasuser.admit;

                                        /* This query gets distinct values of actlevel stored in
the variable "act" dynamically, same can be done for unknown no. of values of any variable.*/

select count (distinct actlevel) into: n from sasuser.admit;

                                        /* This query gets the total count of distinct values of
variable actlevel, which acts as the terminating count for the counter in the loop  */

quit;

%put &act &n;

%do i=1 %to &n;
%let ac = %scan(&act,&i,"@");

data &ac;
```

```
set sasuser.admit;
where actlevel="&ac";
run;

Proc export data=&ac outfile= "C:\Documents and
Settings\sasadm\Desktop\2\admit1.xls"
DBMS=excel replace;
Sheet="&ac";
run;

%end;
%mend exporty;

%exporty;
```

Explained: This macro further automates the processing by making counter in the loop dynamic, similarly for the distinct values of variable actlevel. Dynamic declaration adjusts to any random change in the variable values / variable count, like here Mody, NA is added.

Output / Log display:

```
NOTE: PROCEDURE SQL used (Total process time):
      real time           0.31 seconds
      cpu time            0.28 seconds


HIGH@LOW@MOD@MODY@NA        5
```

Prog 3

```
%macro exporty (d=,l=,v=);

proc sql noprint;
select distinct &v into: act separated by "@" from &l..&d;
select count(distinct &v) into: n from &l..&d;
quit;

%put &act &n;

%do i=1 %to &n;
%let ac=%scan(&act,&i,"@");

data &ac;
set &l..&d;
where &v="&ac";
run;

proc export data=&ac outfile="C:\Documents and
Settings\sasadm\Desktop\2\&d..xls"
dbms=excel replace;
sheet="&ac";
run;

%end;
%mend exporty;

%exporty (d=cargorev,l=sasuser, v=route);
```

Explained: Further automating the macro with library_name, dataset_name, and the variable_name passed as parameters this creates 1 excel file with different sheets based on different routes.

Output / Log display:

```
130  %exporty (d=cargorev,l=sasuser, v=route);
NOTE: PROCEDURE SQL used (Total process time):
      real time           0.23 seconds
      cpu time            0.12 seconds


Route1@Route2@Route3@Route4@Route5@Route6@Route7        7
```

Prog 4

```
%macro exporty (f=,l=,v=);

proc sql noprint;
select distinct &v into: act separated by "@" from &l..&f;
select count(distinct &v) into: n from &l..&f;
quit;

%put &act &n;

%do i=1 %to &n;
%let ac=%scan(&act,&i,"@");

data &ac;
set &l..&f;
where &v="&ac";
run;

proc export data=&ac outfile="C:\Documents and
Settings\sasadm\Desktop\2\&ac..xls"
dbms=excel replace;
sheet="&ac";run;

%end;
%mend exporty;

%exporty (f=cars,l=sashelp, v=make);
```

Explained: This macro creates multiple excel files as per distinct values of variable, here "make", passed as parameter. Also, the dataset name, resulting excel file and the tab name within the excel has the same name as the distinct value of the variable.

Prog 5

```
%macro ffawsm;

proc sql noprint;
select distinct state into: s separated by "@" from
sasuser.frequentflyers;
select count(distinct state) into: n from sasuser.frequentflyers;
quit;

%put &s &n;
%do i=1 %to &n;
```

```
%let st=%scan(&s,&i,"@");

proc sql;
create table &st as
select membertype,count(*) as count from sasuser.frequentflyers
where state="&st" group by membertype;
quit;

proc export data = &st outfile="C:\Documents and
Settings\sasadm\Desktop\2\ff.xls"
dbms=excel replace;
sheet="&st";
run;

%end;

%mend ffawsm;

%ffawsm;
```

Explained: This creates an excel, "ff.xls", with state-wise sheets/tabs having member-type (variable) count in it (portfolio analysis)

Prog 6

```
%macro ffawsm;

proc sql noprint;
select distinct state into: s separated by "@" from
sasuser.frequentflyers;
select count(distinct state) into: n from sasuser.frequentflyers;
quit;

%put &s &n;
%do i=1 %to &n;
%let st=%scan(&s,&i,"@");

proc sql;
select count(*) into: c from sasuser.frequentflyers where state="&st";
quit;

%let c=%sysfunc(compress(&c));

proc sql;
create table &st._&c as select * from sasuser.frequentflyers where
state="&st";
quit;

proc export data = &st._&c outfile="C:\Documents and
Settings\sasadm\Desktop\2\ff2.xls"
dbms=excel replace;
sheet="&st.&c";
run;

%end;
%mend ffawsm;
```

```
%ffawsm;
```

Explained: This creates an excel file “ff2.xls” which has state-wise sheets having name, as value from “&st” (state) concatenated with value from “c” (count, with reference to any state)

Output / Log display:

```
246  %ffawsm;
NOTE: PROCEDURE SQL used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds


AR@AZ@CA@CO@CT@DC@FL@GA@IA@IL@IN@KS@LA@MA@MD@ME@MI@NC@NE@NH@NJ@NY@OH@OK@OR@PA@PQ@QU@TN@TX@V
A@WA@WI       33
NOTE: PROCEDURE SQL used (Total process time):
      real time           0.01 seconds
      cpu time            0.01 seconds


NOTE: Table WORK.AR_5 created, with 5 rows and 11 columns.

NOTE: PROCEDURE SQL used (Total process time):
      real time           0.01 seconds
      cpu time            0.01 seconds
```