



IS03: The Power of PROC SQL

PhUSE 2012

Amanda Reilly
ICON, Dublin, Ireland
15/10/2012

- **Built-in procedure in SAS**
 - SAS implementation of SQL
 - Very powerful tool for improving efficiency
- **Can** - sort, subset, merge datasets
 - create new variables and tables
 - summarize data **all at once!**
- **No Need for Sorting!**
 - Joins/Functions can be performed on unsorted data
- **Efficient code can reduce the overall volume of your program**
 - Improves clarity and legibility
 - Comments pinpoint exactly what each step is doing
- **Quick Validation**

Section 1:

Getting familiar with PROC SQL

```
proc sql <OPTIONS>;  
create table <NewDataset> AS  
  select <distinct> <column names>  
    from <OldDataset>  
      <where>  
      <group BY>  
      <order BY>;  
quit;
```

nodupkey

Creating a new dataset

Variables in the new dataset

Where to take data from

*I.e. Summarising by treatment group,
Built-in Proc Sort*

```
proc sql <OPTIONS>;  
create table <NewDataset> AS  
  select <distinct><column names>  
    from <OldDataset>  
      <group BY>  
      <having>;  
quit;
```

Restricts the queries results

Syntax: Base SAS vs. PROC SQL

Clinical Research

Base SAS

by TRT VARIABLE LEVEL

where trtsdt <= adt <= trtedt

run;

where subjid=""; / subjid=.;
var1= " " ; / var1= . ;

data NewDataset ;

set OldDataset ;

if **then** ;

run ;

PROC SQL

group by TRT, VARIABLE, LEVEL

where adt between trtsdt **and** trtedt

quit;

where subjid is missing

insert into DSET (var1) value (null)

proc sql <OPTIONS>;

create table <NewDataset> AS

select <distinct> <column names>

from <OldDataset>

<**where**>

<**group BY**>

<**order BY**> ;

quit ;

Section 2:

PROC SQL – compact and concise!

Syntax: SELECT statement

Clinical Research

- 'select *' - keeps all variables in a dataset
- Distinct
 - unique records only; count(distinct subjid)
- Fⁿ(var1) as var2:
 - count(distinct subjid) as N
 - mean(X) as average, std(X) as stddev, min(X) as minimum...
 - Rename: 'var1 as var2'
- Set formats and labels:
 - count(distinct SUBJID) as N format=8. label='Total'
 - min(X) as minimum format=\$20. label='Min. Value'
- Retain
 - Variables appear in same order as they are listed

Syntax: Other

Clinical Research

- **Wildcard - %**
 - X% begins with X
 - %X ends with X
 - %X%: Contains X (i.e.. where aeterm = '%pain%')
- **'group by' – computation over subgroups**
 - Number of subjects in each treatment:

```
select trt, count (distinct subjid) as N from olddata
group by trt;
```
- **'order by'**
 - Orders the output (eliminates need to follow data-step with a proc sort)
- **INSERT statement**

```
proc sql;
    insert into NewDataset (var1, var2, var3)
variables: select var4, var5, var6 from OldDataset;
values:     values (3, 5, 'XXX');
quit;
```

Syntax: Example

Variable	Level	Treatment	Frequency Count
AGEGRP	<45 years		2
AGEGRP	<45 years	PLACEBO	14
AGEGRP	<45 years	TREATMENT A	25
AGEGRP	<45 years	TREATMENT B	11
AGEGRP	>65	PLACEBO	0
AGEGRP	>65	TREATMENT A	1
AGEGRP	>65	TREATMENT B	0
AGEGRP	>=45 to <=65 years		3
AGEGRP	>=45 to <=65 years	PLACEBO	13
AGEGRP	>=45 to <=65 years	TREATMENT A	34
AGEGRP	>=45 to <=65 years	TREATMENT B	16
RACE	ASIAN	PLACEBO	0
RACE	ASIAN	TREATMENT A	0
RACE	ASIAN	TREATMENT B	0
RACE	BLACK	PLACEBO	0
RACE	BLACK	TREATMENT A	0
RACE	BLACK	TREATMENT B	0
RACE	OTHER	PLACEBO	0
RACE	OTHER	TREATMENT A	0
RACE	OTHER	TREATMENT B	0
RACE	WHITE		5
RACE	WHITE	PLACEBO	27
RACE	WHITE	TREATMENT A	60
RACE	WHITE	TREATMENT B	27

Demographic table -

Syntax: Example

Clinical Research

Variable	Level	Treatment	Frequency Count
AGEGRP	<45 years		2
AGEGRP	<45 years	PLACEBO	14
AGEGRP	<45 years	TREATMENT A	25
AGEGRP	<45 years	TREATMENT B	11
AGEGRP	>65	PLACEBO	0
AGEGRP	>65	TREATMENT A	1
AGEGRP	>65	TREATMENT B	0
AGEGRP	>=45 to <=65 years		3
AGEGRP	>=45 to <=65 years	PLACEBO	13
AGEGRP	>=45 to <=65 years	TREATMENT A	34
AGEGRP	>=45 to <=65 years	TREATMENT B	16
RACE	ASIAN	PLACEBO	0
RACE	ASIAN	TREATMENT A	0
RACE	ASIAN	TREATMENT B	0
RACE	BLACK	PLACEBO	0
RACE	BLACK	TREATMENT A	0
RACE	BLACK	TREATMENT B	0
RACE	OTHER	PLACEBO	0
RACE	OTHER	TREATMENT A	0
RACE	OTHER	TREATMENT B	0
RACE	WHITE		5
RACE	WHITE	PLACEBO	27
RACE	WHITE	TREATMENT A	60
RACE	WHITE	TREATMENT B	27

Demographic table -

Want to calculate:

Total number of subjects (N)

Number & % in each group

Syntax: Example (PROC SQL)

Clinical Research

N

```
proc sql;
    create table totn as
    select variable, level, trt, count, sum(count) as N
        from demog
    group by variable;
quit;
```

Total n & %

```
proc sql;
    create table total as
    select variable, level, trt, count, sum(count) as LevelCount,
        ((sum(count) / N)* 100/1) as LevelPC format=8.2, N
        from totn
    where trt ne "
    group by level
    order by variable, level, trt;
quit;
```

← *Outputs a sorted dataset*

Syntax: Example (PROC SQL)

Clinical Research

N

```
proc sql;
  create table totn as
  select variable, level, trt, count, sum(count) as N
    from demog
  group by variable;
quit;
```

Total n & %

```
proc sql;
  create table total as
  select variable, level, trt, count, sum(count) as LevelCount,
  ((sum(count) / N)* 100/1) as LevelPC format=8.2, N
    from totn
  where trt ne "
  group by level
  order by variable, level, trt;
quit;
```

← *Outputs a sorted dataset*

Syntax: Example (SAS)

N

```
proc sort data=DEMOG;  
by VARIABLE LEVEL;  
run;
```

```
proc transpose data=DEMOG  
out=DEMOG_T1;  
var LEVEL TRT COUNT;  
by VARIABLE;  
run;
```

```
data TOTN;  
set DEMOG_T1;  
N=sum(COL1-COL13);  
where _NAME_='COUNT';  
keep VARIABLE _NAME_ _LABEL_ N;  
run;
```

Total n & % (1)

```
proc transpose data=DEMOG  
out=DEMOG_T2;  
var TRT COUNT;  
by VARIABLE LEVEL;  
where TRT ne " ";  
run;
```

```
proc sort data=DEMOG_T2;  
by VARIABLE _NAME_ _LABEL_;  
run;
```

```
proc sort data=TOTN;  
by VARIABLE _NAME_ _LABEL_;  
run;
```

Total n & % (2)

```
data SUM;  
format LPC 8.2;  
merge DEMOG_T2 TOTN;  
by VARIABLE _NAME_ _LABEL_;  
LCOUNT=sum(COL1:COL3);  
LPC=(LCOUNT/ N)* 100/1;  
where _NAME_='COUNT';  
run;
```

```
data TOTAL;  
merge SUM DEMOG  
(where=(TRT ne "));  
by VARIABLE LEVEL;  
run;
```

Syntax: Example

	Variable	Level	Treatment	Frequency Count	LCOUNT	LPC	N
1	AGEGRP	<45 years	PLACEBO	14	50	42.02	119
2	AGEGRP	<45 years	TREATMENT A	25	50	42.02	119
3	AGEGRP	<45 years	TREATMENT B	11	50	42.02	119
4	AGEGRP	>65	PLACEBO	0	1	0.84	119
5	AGEGRP	>65	TREATMENT A	1	1	0.84	119
6	AGEGRP	>65	TREATMENT B	0	1	0.84	119
7	AGEGRP	>=45 to <=65 years	PLACEBO	13	63	52.94	119
8	AGEGRP	>=45 to <=65 years	TREATMENT A	34	63	52.94	119
9	AGEGRP	>=45 to <=65 years	TREATMENT B	16	63	52.94	119
10	RACE	ASIAN	PLACEBO	0	0	0.00	119
11	RACE	ASIAN	TREATMENT A	0	0	0.00	119
12	RACE	ASIAN	TREATMENT B	0	0	0.00	119
13	RACE	BLACK	PLACEBO	0	0	0.00	119
14	RACE	BLACK	TREATMENT A	0	0	0.00	119
15	RACE	BLACK	TREATMENT B	0	0	0.00	119
16	RACE	OTHER	PLACEBO	0	0	0.00	119
17	RACE	OTHER	TREATMENT A	0	0	0.00	119
18	RACE	OTHER	TREATMENT B	0	0	0.00	119
19	RACE	WHITE	PLACEBO	27	114	95.80	119
20	RACE	WHITE	TREATMENT A	60	114	95.80	119
21	RACE	WHITE	TREATMENT B	27	114	95.80	119

- A similar procedure might be used to examine the data at a treatment level

- **SAS-**
 - Use a set / merge statement
 - Data must be sorted
- **PROC SQL-**
 - Use a 'join' statement
 - Data doesn't need to be sorted first
 - Benefits
 - Reduction in number of steps needed
 - Possible to join when key variables have different names
 - Join multiple datasets on different levels
- **Note:**
 - Merge: overlays common variables
 - Join: doesn't overlay common variables

Combining Datasets: Join vs. Merge

Clinical Research

- Inner Join 

```
proc sql;
  create table newdata as
  select *
  from dset1 A, dset2 B
  where A.subjid=B.subjid and A.visit=B.visit; quit;
```

```
data newdata;
  merge dset1 dset2;
  by subjid visit;
  if a and b;
run;
```

- Left Outer  / Right Outer  / Full Joins 

```
proc sql;
  create table newdata as
  select *
  from dset1 A LEFT JOIN dset2 B
  [ / RIGHT JOIN / FULL JOIN ]
  on A.subjid=B.subjid and A.visit=B.visit;
quit;
```

```
data newdata;
  merge dset1(in=a) dset2(in=b);
  by subjid visit;
  if a;
  [ / if b / if a or b ]
run;
```

-  Add where B.subjid is null

if a and not b;

Combining Datasets: Cartesian Product

Clinical Research

- Omit 'on' / 'where' statement on a JOIN
- Very useful for creating dummy datasets!

Example:

12 subjects monitored over 9 visits (baseline -> Day 8)

Some subjects missing observations

Want each subject lined up with every visit

- **proc sql;**
 create table dummy as
 select distinct a.subjid, b.visit
 from subj a, visit b
 order by subjid, visit;
quit;

Combining Datasets: Cartesian Product

Clinical Research

- Omit 'on' / 'where' statement on a JOIN
- Very useful for creating dummy datasets!

Example:

12 subjects monitored over 9 visits (baseline -> Day 8)
Some subjects missing observations
Want each subject lined up with every visit

- **proc sql;**
 create table dummy as
 select distinct a.subjid, b.visit
 from subj a, visit b
 order by subjid, visit;
quit;

- Now every subject is matched with every possible visit!

	SUBJID	VISIT
1	11033001	Baseline
2	11033001	Day 1
3	11033001	Day 2
4	11033001	Day 3
5	11033001	Day 4
6	11033001	Day 5
7	11033001	Day 6
8	11033001	Day 7
9	11033001	Day 8
10	11033003	Baseline
11	11033003	Day 1
12	11033003	Day 2
13	11033003	Day 3
14	11033003	Day 4
15	11033003	Day 5
16	11033003	Day 6
17	11033003	Day 7
18	11033003	Day 8
19	11044002	Baseline
20	11044002	Day 1
21	11044002	Day 2

Combining Datasets: Coalesce function

Clinical Research

- A JOIN doesn't overlay common variables like a MERGE does
 - LEFT JOIN: select * on SUBJID - Log Error
Resolution: Rename one of the SUBJID variables (i.e. SUBJID2)
 - FULL JOIN: Use the COALESCE function
 - Picks first non-missing value & overlays the variable you are joining by

- Example:

7 subjects in DM dataset, 7 in PD dataset

1 subject in each is missing from the other.

Want a full join so we can see all 8 subjects in the new dataset.

```
proc sql;
  create table newdata as
  select a.*, b.*
  from dset1 A FULL JOIN dset2 B
  on A.subjid=B.subjid ;
quit;
```

Combining Datasets: Coalesce function

Clinical Research

DM and PD data:

SUBJID	TRT
BU001IM-31-010	Treatment #1
BU001IM-31-011	Placebo
BU001IM-31-012	Treatment #2
BU001IM-31-013	Treatment #2
BU001IM-31-014	Treatment #1
BU001IM-31-015	Treatment #2
BU001IM-31-016	Placebo

SUBJID	TRT	Any Major Protocol deviation?
BU001IM-31-010	Treatment #1	No
BU001IM-31-011	Placebo	No
BU001IM-31-012	Treatment #2	Yes
BU001IM-31-014	Treatment #1	No
BU001IM-31-015	Treatment #2	No
BU001IM-31-016	Placebo	No
BU001IM-31-017	Treatment #1	Yes

FULL JOIN

SUBJID	TRT	Race	Sex	Age Group	Any Major Protocol deviation?
BU001IM-31-010	Treatment #1	WHITE	F	<45 years	No
BU001IM-31-011	Placebo	WHITE	F	<45 years	No
BU001IM-31-012	Treatment #2	WHITE	F	>=45 to <=65 years	Yes
BU001IM-31-013	Treatment #2	WHITE	M	<45 years	
BU001IM-31-014	Treatment #1	WHITE	F	<45 years	No
BU001IM-31-015	Treatment #2	WHITE	M	<45 years	No
BU001IM-31-016	Placebo	WHITE	F	<45 years	No
					Yes

ID and Treatment of subject 31-017 is missing since the join didn't overlay the SUBJID / TRT

Combining Datasets: Coalesce function

Clinical Research

```
proc sql;  
  create table newdata as  
  select  
    coalesce (a.subjid, b.subjid) as SUBJID,  
    coalesce (a.trt, b.trt) as TRT,  
    race, sex, agegrp, mpdev  
  from dset1 A FULL JOIN dset2 B  
  on A.subjid=B.subjid;  
quit;
```

Combining Datasets: Coalesce function

Clinical Research

```
proc sql;
```

```
  create table newdata as
```

```
  select
```

```
    coalesce (a.subjid, b.subjid) as SUBJID,
```

```
    coalesce (a.trt, b.trt) as TRT,
```

```
    race, sex, agegrp, mpdev
```

```
  from dset1 A FULL JOIN dset2 B
```

```
  on A.subjid=B.subjid;
```

```
quit;
```

	SUBJID	TRT	Race	Sex	Age Group	Any Major Protocol deviation?
1	BU001IM-31-010	Treatment #1	WHITE	F	<45 years	No
2	BU001IM-31-011	Placebo	WHITE	F	<45 years	No
3	BU001IM-31-012	Treatment #2	WHITE	F	>=45 to <=65 years	Yes
4	BU001IM-31-013	Treatment #2	WHITE	M	<45 years	
5	BU001IM-31-014	Treatment #1	WHITE	F	<45 years	No
6	BU001IM-31-015	Treatment #2	WHITE	M	<45 years	No
7	BU001IM-31-016	Placebo	WHITE	F	<45 years	No
8	BU001IM-31-017	Treatment #1				Yes



Section 3:

Optimizing the use of PROC SQL

Example:

Want to know number of subjects are in each treatment group to include this number in each treatment column header

	Placebo (N=47)	Treatment 1 (N=48)	Treatment 2 (N=49)	Total (N=144)
n	X	X	X	X
Mean	X	X	X	X
SD	X	X	X	X
Minimum	X	X	X	X
Median	X	X	X	X
Maximum	X	X	X	X

We want to calculate this and then save it in a variable which we can call in when needed

```
proc sql noprint;  
  select count(distinct trtan) into :trtcnt  
  from SL;
```

Number of treatments

```
%do i = 1 %to &trtcnt;  
  select distinct trta into :trtn&i  
  from SL  
  where trtan = &i;
```

*Loop from 1 to trtcnt; creates trtn1,...,trtntrtcnt
Treatment names for each macro variable*

```
  select count(distinct SUBJID) into :trtcnt&i  
  from SL  
  where trtan = &i;
```

Number of subjects in each treatment

```
%end;  
quit;
```

****resolving the macro to know the count****;

```
%do i = 1 %to &trtcnt;  
  %put trtn&i:&&trtn&i trtcnt&i:&&trtcnt&i;  
%end;  
%put trtcnt:&trtcnt;
```

```
proc sql noprint;
  select count(distinct trtan) into :trtcnt
  from SL;
```

Number of treatments

```
%do i = 1 %to &trtcnt;
  select distinct trta into :trtn&i
  from SL
  where trtan = &i;
```

*Loop from 1 to trtcnt; creates trtn1,...,trtntrtcnt
Treatment names for each macro variable*

```
  select count(distinct SUBJID) into :trtcnt&i
  from SL
  where trtan = &i;
```

Number of subjects in each treatment

```
%end;
quit;
```

****resolving the macro to know the count****;

```
%do i = 1 %to &trtcnt;
  %put trtn&i:&&trtn&i trtcnt&i:&&trtcnt&i;
%end;
%put trtcnt:&trtcnt;
```



```
define trt1 / ... "1st Treatment" (N=&trt1) " ... ;
define trt2 / ... "2nd Treatment" (N=&trt2) " ... ;
define trt3 / ... "3rd Treatment" (N=&trt3) " ... ;
```

Joins: Joining Subsets / Self Join

Clinical Research

- Take a subset of one dataset and join that with another dataset, all in one step!
- Example: Comparing Baseline data with data from other visits
 - Subset the data down to those with baseline flag = Y
 - Compare those records with all other records
 - Self-join: both A and B are the same dataset!

proc sql;

```
create table newdata as
select a.*, b.base, a.aval - b.base as basechange
from dset1 A LEFT JOIN (select usubjid, aval as base from dset1 where ablfl='Y') as B
on a.usubjid=b.usubjid
order by usubjid, avisitn;
```

quit;

- Only compare visits after baseline?
on a.usubjid=b.usubjid and avisitn >1

Joins: 'Fuzzy' conditional Merge

Clinical Research

- Normally: Merge datasets on one/more common variables with exact matching values (i.e. Identical USUBJID in both)
- 'Fuzzy' Merge doesn't need to values to be exact, but can search on a condition
- Example:
To find out which dose a subject was taking at the time of a particular adverse event.

```
proc sql;
  create table newdata as
  select a.*, b.exdose as AEDOSE
  from ADAE A
  LEFT JOIN
  (select usubjid, exstdt, exenddt, exdose from ADEX) as B
  on a.usubjid=b.usubjid and aestdt between exstdt and exenddt;
quit;
```

Joins: Joining Multiple Datasets

Clinical Research

- Merging multiple datasets on different levels:

SAS:

- Sort all datasets being merged on the first level
- Merge those datasets in a data step
- Sort that dataset & the datasets being merged on the second level
- Merge those datasets in a data step... etc

- PROC SQL:** - Just one step!

- Example:

Need to join ADaM.ADSL and SDTM.VS on USUBJID, and merge this with SDTM.SV on VISITNUM. USUBJID is common to all three, but VISITNUM is only common to B and C, not A.

```
proc sql;
  create table newdata as
  select a.*, b.*, c.* from
    (select * from ADAM.ADSL ) as A,
    (select usubjid, visitnum, vstestcd, vsstresn from SDTM.VS) as B,
    (select usubjid, visitnum, svstdtc from SDTM.SV ) as C
  where A.usubjid=B.usubjid and B.usubjid=C.usubjid
  and B.visitnum=C.visitnum;
quit;
```

- Lack of understanding -> possibly incorrect results / not what expecting
- Care needed with
 - Full joins (coalesce function)
 - Many-to-many joins (needs care to ensure accuracy)
- When to use PROC SQL/SAS?
 - Depends on
 - size of datasets
 - relationship between them
 - sparseness/density of the matches
 - Simple join: Merge statement is more efficient (CPU time)
 - Large/unsorted data: Inner Join
 - Many-to-many merge: proc sql is the preferred choice
- Efficiency in run-time/program space vs. time taken to write the code
 - How often will the code be run?

- Just one step can do so much!
 - Concise code can improve clarity & legibility
- Care is needed with the more complex joins
- Very worthwhile getting familiar with PROC SQL syntax
 - Very useful for
 - Basic statistical functions
 - Macro variables
 - Flexibility of joins (& no preprocessing)
 - Ability to edit / update another programmers PROC SQL code

- Comments and questions are welcome to

Author Name: Amanda Reilly

Company: ICON PLC

Email: Amanda.Reilly@iconplc.com

Web: <http://www.iconplc.com/>

Questions?

