DH02
Consistent Data Delivery

Dr. Thevaki Mahadevan
Consistent Data Delivery - Introduction

- Inconsistencies in the raw data, between data delivery
- Data issues during mapping or transfer process
- Biostatistics best approaches
  - Manually checks all issues, including subject counts in key datasets
  - Write SAS code to check each delivery data
- The advantages of programmatic checks
  - more reliable
  - minimizes human error
  - less time consuming in the long run.
- A program can bulk check all raw datasets, variables and observations that are delivered each time.
Consistent Data Delivery - Typical findings

- External data issue
- Reference ranges changed
- Conversion formulas changed
- Code/decode values changed
- Datasets not delivered
- Missing variables/datasets.
- Extra variables
- Extra datasets
Consistent Data Delivery- Processes and Solutions

- Programmers concentrate on current data issues.
- Very difficult to spot these issues manually by stat/lead reviews.
- Check the data differences using programs.
- When datasets are saved in different folders, i.e. libraries, they can be referenced by libnames.

/* Location of Old DM data with the new delivered DM data*/
libname DMOLD "Volumes/app/sites/project/ICON_derm/protocol/data/data_old"; /*Old DM data*/
libname DMNEW "Volumes/app/sites/project/ICON_derm/protocol/data/data_old"; /*New DM data*/

Figure 1: Sample sashelp.vcolumn dataset
Consistent Data Delivery - Meta data and dataset Differences

A merge on datasets

/*Get list of datasets and variables names from both libnames by accessing Dictionaries*/
data array;
  set asshelp.xcolumm (keep-libname memname name length label type format inform);
  where libname in ('DMOLD', 'DMNEW');
run;

/*Check which datasets are in both library and if missing from one area*/
proc sort data-array out-du(drop-libname) nodupkey; where libname='DMOLD'; by memname; run;
proc sort data-array out-du(drop-libname) nodupkey; where libname='DMNEW'; by memname; run;

data both data(keep-memname) incld_data innew_data ;
  merge dmol(L=e) dm(1=0);
  by memname;
  if a and b then output both_data;
  if a and not b then output incld_data;
  if b and not a then output innew_data;
run;

<table>
<thead>
<tr>
<th>MEMNAME</th>
<th>IN</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALLERGY</td>
<td>inold_dat</td>
</tr>
<tr>
<td>LAB_SDW</td>
<td>inold_dat</td>
</tr>
<tr>
<td>LB</td>
<td>inold_dat</td>
</tr>
<tr>
<td>NKS</td>
<td>inold_dat</td>
</tr>
<tr>
<td>OMIC</td>
<td>inold_dat</td>
</tr>
</tbody>
</table>

Figure 2: Example of output showing datasets only in old data
Consistent Data Delivery - Meta data and variable Differences

A merge on datasets and variables

```sas
/* Check which datasets and variables are in both library and if missing from one area */
proc sort data=old out=old(drop=library); where libname='DMOLO'; by _merge name; run;
proc sort data=old out=old(drop=library); where libname='DMOHE'; by _merge name; run;

data both var inold_var innew_var;
  merge dmy(lib=a) dmy(lib=b);
  by _merge name;
  if a and b then output both_var;
  if a and not b then output inold_var;
  if b and not a then output innew_var;
run;
```

<table>
<thead>
<tr>
<th>MIMENAME</th>
<th>NAME</th>
<th>TYPE</th>
<th>LENGTH</th>
<th>LABELS</th>
<th>FORMAT</th>
<th>INFORMAT</th>
<th>IN</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADVERSE</td>
<td>DRSNYTF</td>
<td>char</td>
<td>200</td>
<td>Body Site Involvement-FOL</td>
<td>$200.</td>
<td>$200.</td>
<td>inold_var</td>
</tr>
<tr>
<td>ADVERSE</td>
<td>DRSNYTE</td>
<td>char</td>
<td>200</td>
<td>Body Site Involvement-IV</td>
<td>$200.</td>
<td>$200.</td>
<td>inold_var</td>
</tr>
<tr>
<td>ADVERSE</td>
<td>DRSNYTN</td>
<td>num</td>
<td>8</td>
<td>Body Site Involvement-DIV</td>
<td>10.</td>
<td></td>
<td>inold_var</td>
</tr>
</tbody>
</table>

Figure 3: Example of output showing only variables in old dataset

A macro variable referencing to call common dataset

```sas
/*
proc compare DM old with DM new
/* Create macro variables for each common datasets found in both library and for total number / of datasets found. /
/*
proc sort data=both data out= both dat1 nodupkey; by _name; run;
data _null_
  and both dat1 end-last;
  if last then call symput('tot1', n_);
  call symput('DAT1'|left(_N_), trim(left(_name)));
run;
```
Consistent Data Delivery - Bulk Proc compare macro

- All numeric variables selected as by variable
- Conditional key by variable
- Conditional where statement

BULK RUN OF PROC COMPARE

%macro cmp:
  %do i = 1 %to stots:
    data byvar (keep=mmame byvar byver2 nonel);
    set bshs_var;
    retain byvar byvar2 nonel;
    format byvar2 $15., bbyar $9.00, nonel $8.;
    by mmame;
    if strip(type) eq 'num' then do;
    if first:mmame then byvar-name;
    else byvar=cat('",byvar,name);"
    end;
    if first:mmame then byvar2="";
    if none eq 'SUBJID' then byvar2='study subjid';
    if first:mmame then none="";
    if none eq 'NONEL' then nonel='nonel';
    if last:mmame then output:
      run;
    data _null_
    retain byvar;
    if _n_
    if strip(mmame) eq "&mmame" then call symput ('byvar' || left(_n_), trim(left(byvar)));
    if strip(mmame) eq "&mmame" then call symput ('byvar2' || left(_n_), trim(left(byvar2)));
    if strip(mmame) eq "&mmame" then call symput ('non' || left(_n_), trim(left(nonel)));
    run;
  %end:
Consistent Data Delivery - Bulk Proc compare macro

- Datasets sorted before proc compare
- Proc means kept within the macro

```sas
PROC SORT DATA=dataset1 out=sorted1;
by %if index(subvar1, subjid) %then $do; subvar1 subvar2; %end;
%else $do; subvar1; %end;
%if index(index1, index2) %then $do;
where index1 ne 'NONE'; %end;
run;

PROC SORT DATA=dataset2 out=sorted2;
%if index(index2, index1) %then $do;
where index1 ne 'NONE'; %end;
by %if index(subvar2, subjid) %then $do; subvar2 subvar1; %end;
%else $do;subvar1; %end;
run;

PROC COMPARE DATA=sorted1 compare=sorted2 LISTALL NOVALUES OUT=diffs OUTNOVALUE);
id %if index(subvar2, subjid) %then $do; subvar2 subvar1; %end;
%else $do;subvar1; %end;
run;

PROC MEANS DATA=sorted1 n missing;
run;
%end;
%end;
%comp;
```
Consistent Data Delivery - Proc compare output

- A top level dataset summary with total number of obs and vars present in the two datasets.

- The variable summary with variables differences found between the two datasets.

```
proc compare data=work.Universe.a out=compare dataset1=work.Universe bdataset2=work.Universe.b
run;
```

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Created</th>
<th>Modified</th>
<th>Which</th>
<th>Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>work.Universe</td>
<td>11/04/11 08:02</td>
<td>11/04/11 08:02</td>
<td>02</td>
<td>1622</td>
</tr>
<tr>
<td>work.Universe.b</td>
<td>11/04/11 08:30</td>
<td>11/04/11 08:30</td>
<td>30</td>
<td>2013</td>
</tr>
</tbody>
</table>

**Variable Summary**

- Number of variables in common: 49.
- Number of variables in work.Universe but not in work.Universe.b: 0.
- Number of variables in work.Universe.b but not in work.Universe: 1.
- Number of different attributes: 1.
- Number of different variables: 29.

**Listing of Variables in work.Universe but not in work.Universe.b**

```
Variable Type | Length | Format | Inform  | Label
--------------|--------|--------|---------|-------
OBSSITE       |      8 |      8 | Text    | Body Site Involvement-UN
OBSSITE      |      8 |      8 | Text    | Body Site Involvement-UL
OBSSITE      |      8 |      8 | Text    | Body Site Involvement-ULV
```

**Listing of Variables in work.Universe.b but not in work.Universe**

```
Variable Type | Length | Format | Inform | Label
--------------|--------|--------|--------|-------
ACSITE       |      5 | Text   | Text   | AC Site Category
ACSITE      |      5 | Text   | Text   | Body Site Involvement-UN
ACSITE      |      5 | Text   | Text   | Body Site Involvement-UL
ACSITE      |      5 | Text   | Text   | Body Site Involvement-ULV
```
Consistent Data Delivery- Proc compare output

• The observation summary with number of observation differences found between the two datasets.

• Any significant drop in observation count from the old data is of potential concern.

Observation Summary

Number of Observations in Common: 1702.
Number of Observations in WORK.NORTHCREED but not in WORK.SOUTHCREED: 145.
Number of Observations in WORK.SOUTHCREED but not in WORK.NORTHCREED: 476.
Total Number of Observations Read from WORK.NORTHCREED: 1959.
Total Number of Observations Read from WORK.SOUTHCREED: 1536.
Number of Observations with Some Compared Variables Unqual: 8.
Number of Observations with All Compared Variables Equal: 1794.

• The value comparison summary is of much less value at this stage and can be skimmed through.

Value Comparison Summary

Number of Variables Compared with All Observations Equal: 56.
Number of Variables Compared with Some Observations Unqual: 1.
Number of Variables with Missing Value Differences: 1.
Total Number of Values which Compare Unqual: 0.

Variables with Unqual Values

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Len</th>
<th>Label</th>
<th>Error</th>
<th>Warning</th>
<th>Missing</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1NMTT</td>
<td>CHARS</td>
<td>80</td>
<td>Action Placement Order Test</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>A1ECOL</td>
<td>CHAR</td>
<td>100</td>
<td>Lowest Level Test (ALT)</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>
Consistent Data Delivery- Proc compare on selected dataset

Individual detail dataset comparison

```
/*
  when only need to run new dataset use below proc compare
*/

%let dataset=adverse;
%let by-union access collate id=union:
/*
  proc sort data=adold; dataset out=addataset;
  by id;
  where nonel ne 'NONE';
  run;
*/

proc sort data=adnew; dataset out=addataset;
   where nonel ne 'NONE';
   run;

proc compare data=adataset compare=addataset listbase out=addataset outnonequal;
   id id;
   run;
```

Further merge with tailored key by variables

```
data both_obs inold_obs innew_obs;
  merge addataset (in=a) addataset (in=b);
  by id;
  if a and b then output both_obs;
  if a and not b then output inold_obs;
  if b and not a then output innew_obs;
  run;
```

<table>
<thead>
<tr>
<th>Subject ID</th>
<th>Lowest Level Term (LLT)</th>
<th>AE Investigator Test - Raw - MedDRA</th>
<th>Adverse Event From Date</th>
<th>Adverse Event To Date</th>
<th>IN</th>
</tr>
</thead>
<tbody>
<tr>
<td>0011014</td>
<td>Abrasions</td>
<td>ARM ABRASIONS</td>
<td>09JUL2014</td>
<td></td>
<td>inold_obs</td>
</tr>
<tr>
<td>0011014</td>
<td>Abrasions</td>
<td>LEFT ARM ABRASIONS</td>
<td>09JUL2014</td>
<td>05AUG2014</td>
<td>innew_obs</td>
</tr>
</tbody>
</table>

Figure 4: Example of output showing the observation differences between the old and new data
Consistent Data Delivery-Missing values

- Missing and non-missing values checked by using proc means.

- Any high number of missing values of concern can be identified this way.

---

The SAS System
The MEANS Procedure

<table>
<thead>
<tr>
<th>Variable</th>
<th>Label</th>
<th>N</th>
<th>Miss</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADJакс</td>
<td>ME Bike Category</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>KDATE</td>
<td>ME Class</td>
<td>240</td>
<td>0</td>
</tr>
<tr>
<td>ACCESS</td>
<td>Accessible To</td>
<td>200</td>
<td>0</td>
</tr>
<tr>
<td>LOCNAME</td>
<td>Login To</td>
<td>200</td>
<td>0</td>
</tr>
<tr>
<td>LOCOPEN</td>
<td>Locally To</td>
<td>200</td>
<td>0</td>
</tr>
<tr>
<td>RECIN</td>
<td>Received DCF Id</td>
<td>200</td>
<td>0</td>
</tr>
<tr>
<td>SHREDID</td>
<td>SHC Sheet Number</td>
<td>200</td>
<td>0</td>
</tr>
<tr>
<td>QUALITYQ</td>
<td>Qualifying Question</td>
<td>200</td>
<td>0</td>
</tr>
</tbody>
</table>

- Spec on required variables can be crossed checked for missing required variables
Consistent Data Delivery-Summary

This programmatic approach will help us to identify most of the differences, including:

- any differences in the variables or datasets
- significant drop in number of observations
- any common patterns seen across the dataset
- concern level of missing observations
- missing variables or datasets

Once these are checked and are accounted for, one can be more confident in the consistency of the data being delivered.
Consistent Data Delivery

Questions?