Data Conversion for Smaller Organisations: Mapping it Easy

Sofia Vale, OCS Life Sciences, ‘s-Hertogenbosch

Introduction

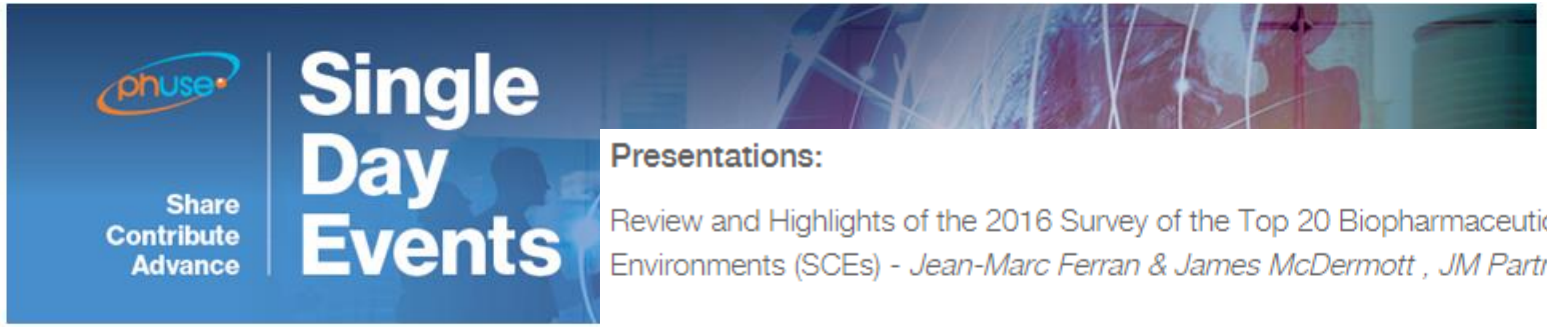
- Clinical data conversion to a standard (e.g. SDTM or ADaM) is not new, but there is still some struggle:
  - Lot of repetitive coding
    - In one program
    - Between data conversion programs for multiple studies
  - Using expensive data conversion software may not be feasible

- In 2016, OCS Life Sciences built and introduced its own internal mapping engine
  - It has made our data conversion easier, faster and more accessible

Mapping Engine

More information, see PhUSE 2017 SDE Utrecht:

https://www.phuse.eu/utrecht-sde-2017



Presentations:

Review and Highlights of the 2016 Survey of the Top 20 Biopharmaceutical Companies' Statistical Computing Environments (SCEs) - *Jean-Marc Ferran & James McDermott , JM Partners*

Blockchain for the Next-level Economy – Ad Kroft, *dutch digital delta*

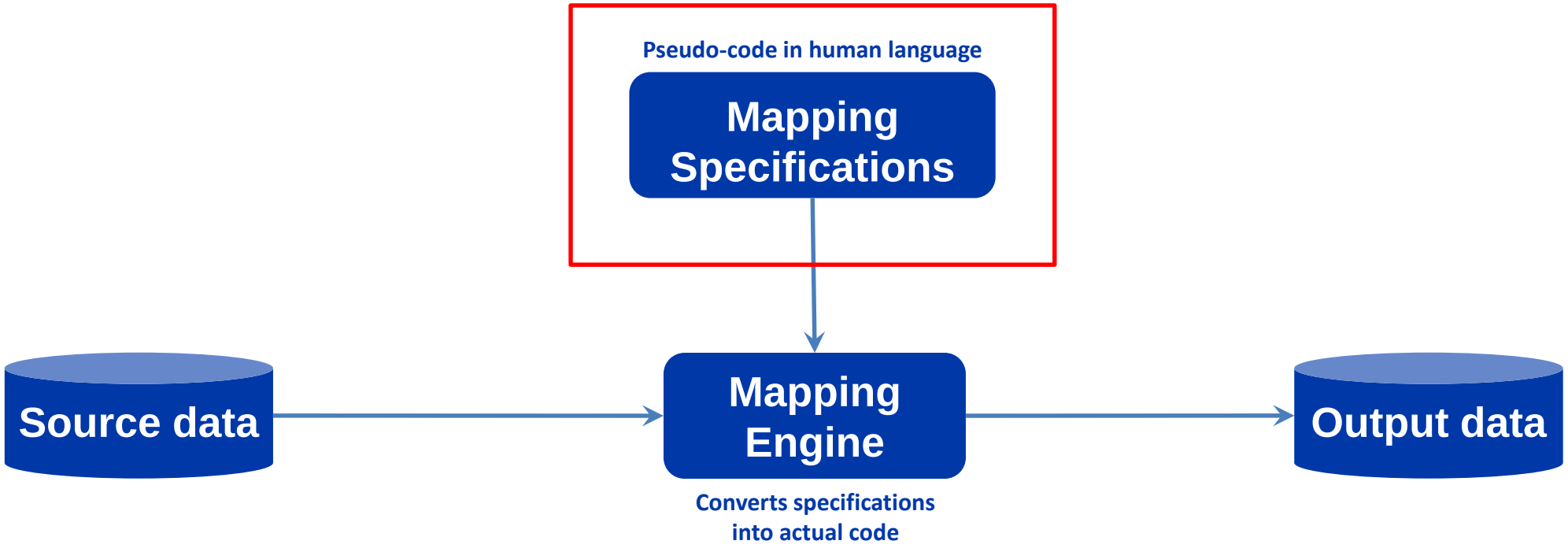
Interoperability, a Real Life Case – Wouter van de Weghe, *SAS & Xavier Deflorenne, GSK*

Create Your Own SDTM Mapping Framework – Bas van Bakel, *OCS Consulting*

"Big Data" Analytics, or the Fallacy behind Risk-based Monitoring – Hartmut Schmied, *Biorasi*

Achieving Interoperability by Automating the Generation of CDISC Artefacts: Design Considerations for an Operational Framework – Andy Richardson, *d-Wise*

Mapping Engine



Mapping Specifications

One file, containing the following tabs:

- Metadata: Defines the dataset, variable, attributes
- Mapping: Contains all specifications and (pseudo-)code
- Recode: Stores all recoding information
- Order: Sets the creation order of output datasets
- Settings: Personalises the process and output

Specifications: Metadata

Metadata | Mapping | Recode | Order | Settings

DOMAIN	DOM_LABEL	VARIABLE	VAR_LABEL	TYPE	LENGTH	FORMAT	UPCASE	SORTVAR
DM	Demographics	STUDYID	Study Identifier	C	200		Y	1
DM	Demographics	DOMAIN	Domain Abbreviation	C	200		Y	
DM	Demographics	USUBJID	Unique Subject Identifier	C	200		Y	2
DM	Demographics	SUBJID	Subject Identifier	C	200			
DM	Demographics	RFSTDTC	Subject Reference Start Date/Time	C	200			
DM	Demographics	DTHDTC	Date/Time of Death	C	200			
DM	Demographics	SEX	Sex	C	200		Y	
VS	Vital Signs	STUDYID	Study Identifier	C	200		Y	1
VS	Vital Signs	DOMAIN	Domain Abbreviation	C	200		Y	
VS	Vital Signs	USUBJID	Unique Subject Identifier	C	200		Y	2
VS	Vital Signs	VSSEQ	Sequence Number	N	8			
VS	Vital Signs	VSTESTCD	Vital Signs Test Short Name	C	200		Y	3
VS	Vital Signs	VSTEST	Vital Signs Test Name	C	200			
VS	Vital Signs	VSORRES	Result or Finding in Original Units	C	200			
VS	Vital Signs	VSORRESU	Original Units	C	200			
VS	Vital Signs	VSSTRESC	Result or Finding in Standard Format	C	200			
VS	Vital Signs	VSSTRESN	Numeric Result/Finding in Standard Unit	N	8	4.2		
VS	Vital Signs	VSDTC	Date/Time of Collection	C	200			
VS	Vital Signs	VSDY	Study Day of Collection	N	8			
VS	Vital Signs	VSBLFL	Baseline Flag	C	200		Y	

Specifications: Mapping

Metadata | **Mapping** | Recode | Order | Settings

SOURCE_DS	SOURCE_VAR	TARGET_DS	TARGET_VAR	SPECIFICATION	FUNCTION
source.demog		DM	STUDYID	Assign value "EX001" for all records	ASSIGN[EX001]
source.demog	SUBJECT	DM	SUBJID	Copy from source variable	COPY
source.demog		DM	USUBJID	Concatenate STUDYID and SUBJID, separated by "-"	CONCAT[STUDYID||"-"||SUBJID]
source.demog	DEMOG_SEX	DM	SEX	Recode according to the recode list GENDER	RECODE[GENDER]
source.demog	DEMOG_DATE	DM	RFSTDTC	Create ISO datetime from the source variables DEMOG_DATE and DEMOG_TIME	ISODATETIME[ISO=DEMOG_DATE, TIME=DEMOG_TIME]
source.demog	DEMOG_TIME	DM	RFSTDTC	See derivation at DEMOG_DATE	
source.demog		DM	DTHDTC	Set to missing	BLANK
source.vitals	OCCUR	VS		Select only the cases where vital signs were collected	WHERE[OCCUR = 1]
source.vitals	SUBJECT_VS	VS	USUBJID	Copy only the characters after the "."	FUNCTION[USUBJID=scan(SUBJECT_VS,2,".");]
source.vitals	HEIGHT	VS	VSTESTCD	For each record generate one "Height" record: Assign value "HEIGHT" for all records	STACK1[#ASSIGN[HEIGHT]#;]
source.vitals	HEIGHT	VS	VSORRES	For each record generate one "Height" record: Copy from source variable	STACK1[#COPY#;]
source.vitals	HEIGHTU	VS	VSORRESU	For each record generate one "Height" record: Copy from source variable when VSORRES is not missing	STACK2[if ^missing(VSORRES) then #COPY#; else #BLANK#;]
source.vitals	WEIGHT	VS	VSTESTCD	For each record generate one "Weight" record: Assign value "WEIGHT" for all records	STACK2[#ASSIGN[WEIGHT]#;]
source.vitals	WEIGHT	VS	VSORRES	For each record generate one "Weight" record: Copy from source variable	STACK2[#COPY#;]
source.vitals	WEIGHTU	VS	VSORRESU	For each record generate one "Weight" record: Copy from source variable when VSORRES is not missing	STACK2[if ^missing(VSORRES) then #COPY#; else #BLANK#;]
source.vitals	EXAM_DT	VS	VSDTC	Create ISO datetime from source ISO datetime	ISODATETIME[ISO=EXAM_DT]
		VS		**POSTSTEP1: Combine source datasets.** Combine mapped_source_vitals with mapped_source_demog, mapped_source_vitals2	POSTSTEP1[data work.vs01; set work.mapped_source_vitals work.mapped_source_demog; run;]
		VS		**POSTSTEP2: Add SDTM DM.RFSTDTC to mapped VS data** Obtain RFSTDTC from OUTLIB.DM by merging the mapped VS data with OUTLIB.DM on USUBJID.	POSTSTEP2[proc sql; create table work.vs02 as select a.*, b.rfstdtc from work.vs01 as A

Specifications: Mapping

Metadata | **Mapping** | Recode | Order | Settings

- Mapping specifications: how to go from the source data to the output data
  - All derivations from source to target
  - Specifications for each derivation

- Reading the map:
  - Step 1: Yellow map → Mapping straight from the source
  - Step 2: Blue map → Mapping adjustments (poststeps)
  - Step 3: (behind the scenes) → Making the final domain

SOURCE_DS	SOURCE_VAR	TARGET_DS	TARGET_VAR	SPECIFICATION	FUNCTION
source.demog		DM	STUDYID	Assign value "EX001" for all records	ASSIGN[EX001]
source.demog	SUBJECT	DM	SUBJID	Copy from source variable	COPY
source.demog		DM	USUBJID	Concatenate STUDYID and SUBJID, separated by "-"	CONCAT[STUDYID]|"-"|[SUBJID]
source.demog	DEMOG_SEX	DM	SEX	Recode according to the recode list GENDER	RECODE[GENDER]
source.demog	DEMOG_DATE	DM	RFSTDTC	Create ISO datetime from the source variables DEMOG_DATE and DEMOG_TIME	ISODATETIME[ISO=DEMOG_DATE, TIME=DEMOG_TIME]
source.demog	DEMOG_TIME	DM	RFSTDTC	See derivation at DEMOG_DATE	
source.demog		DM	DTHDTC	Set to missing	BLANK
source.vitals	OCCUR	VS		Select only the cases where vital signs were collected	WHERE[OCCUR = 1]
source.vitals	SUBJECT_VS	VS	USUBJID	Copy only the characters after the "-"	FUNCTION[USUBJID=scan(SUBJECT_VS,2,"-");]
source.vitals	HEIGHT	VS	VSTESTCD	For each record generate one "Height" record: Assign value "HEIGHT" for all records	STACK1[#ASSIGN[HEIGHT]#;]
source.vitals	HEIGHT	VS	VSORRES	For each record generate one "Height" record: Copy from source variable	STACK1[#COPY#;]
source.vitals	HEIGHTU	VS	VSORRESU	For each record generate one "Height" record: Copy from source variable when VSORRES is not missing	STACK2[if ^missing(VSORRES) then #COPY#; else #BLANK#;]
source.vitals	WEIGHT	VS	VSTESTCD	For each record generate one "Weight" record: Assign value "WEIGHT" for all records	STACK2[#ASSIGN[WEIGHT]#;]
source.vitals	WEIGHT	VS	VSORRES	For each record generate one "Weight" record: Copy from source variable	STACK2[#COPY#;]
source.vitals	WEIGHTU	VS	VSORRESU	For each record generate one "Weight" record: Copy from source variable when VSORRES is not missing	STACK2[if ^missing(VSORRES) then #COPY#; else #BLANK#;]
source.vitals	EXAM_DT	VS	VSDTC	Create ISO datetime from source ISO datetime	ISODATETIME[ISO=EXAM_DT]
		VS		**POSTSTEP1: Combine source datasets.** Combine mapped_source_vitals with mapped_source_demog, mapped_source_vitals2	POSTSTEP1[data work.vs01; set work.mapped_source_vitals work.mapped_source_demog; run;]
		VS		**POSTSTEP2: Add SDTM DM.RFSTDTC to mapped VS data** Obtain RFSTDTC from OUTLIB.DM by merging the mapped VS data with OUTLIB.DM on USUBJID.	POSTSTEP2[proc sql; create table work.vs02 as select a.*, b.rfstdtc from work.vs01 as a

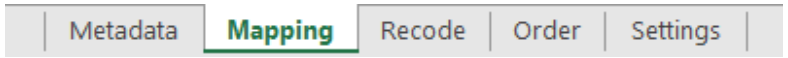
Specifications: Mapping

Metadata | **Mapping** | Recode | Order | Settings

SOURCE_DS	SOURCE_VAR	TARGET_DS	TARGET_VAR	SPECIFICATION	FUNCTION
source.demog		DM	STUDYID	Assign value "EX001" for all records	ASSIGN[EX001]
source.demog	SUBJECT	DM	SUBJID	Copy from source variable	COPY
source.demog		DM	USUBJID	Concatenate STUDYID and SUBJID, separated by "-"	CONCAT[STUDYID||"-"||SUBJID]
source.demog	DEMOG_SEX	DM	SEX	Recode according to the recode list GENDER	RECODE[GENDER]
source.demog	DEMOG_DATE	DM	RFSTDTC	Create ISO datetime from the source variables DEMOG_DATE and DEMOG_TIME	ISODATETIME[ISO=DEMOG_DATE, TIME=DEMOG_TIME]
source.demog	DEMOG_TIME	DM	RFSTDTC	See derivation at DEMOG_DATE	
source.demog		DM	DTHDTC	Set to missing	BLANK
source.vitals	OCCUR	VS		Select only the cases where vital signs were collected	WHERE[OCCUR = 1]
source.vitals	SUBJECT_VS	VS	USUBJID	Copy only the characters after the "."	FUNCTION[USUBJID=scan(SUBJECT_VS,2,".");]
source.vitals	HEIGHT	VS	VSTESTCD	For each record generate one "Height" record: Assign value "HEIGHT" for all records	STACK1[#ASSIGN[HEIGHT]#;]
source.vitals	HEIGHT	VS	VSORRES	For each record generate one "Height" record: Copy from source variable	STACK1[#COPY#;]
source.vitals	HEIGHTU	VS	VSORRESU	For each record generate one "Height" record: Copy from source variable when VSORRES is not missing	STACK2[if ^missing(VSORRES) then #COPY#; else #BLANK#;]
source.vitals	WEIGHT	VS	VSTESTCD	For each record generate one "Weight" record: Assign value "WEIGHT" for all records	STACK2[#ASSIGN[WEIGHT]#;]
source.vitals	WEIGHT	VS	VSORRES	For each record generate one "Weight" record: Copy from source variable	STACK2[#COPY#;]
source.vitals	WEIGHTU	VS	VSORRESU	For each record generate one "Weight" record: Copy from source variable when VSORRES is not missing	STACK2[if ^missing(VSORRES) then #COPY#; else #BLANK#;]
source.vitals	EXAM_DT	VS	VSDTC	Create ISO datetime from source ISO datetime	ISODATETIME[ISO=EXAM_DT]
		VS		**POSTSTEP1: Combine source datasets.** Combine mapped_source_vitals with mapped_source_demog, mapped_source_vitals2	POSTSTEP1[data work.vs01; set work.mapped_source_vitals work.mapped_source_demog; run;]
		VS		**POSTSTEP2: Add SDTM DM.RFSTDTC to mapped VS data** Obtain RFSTDTC from OUTLIB.DM by merging the mapped VS	POSTSTEP2[proc sql;

Source data **Target data** **Human-readable specifications** **Machine-executable (pseudo-)code**

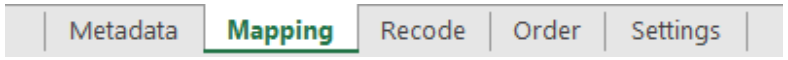
Mapping Engine Functions



- Several engine-specific functions created → pseudo-code
  - Easier to understand
  - Requires less code

STRAIGHTFORWARD	NOT-SO-STRAIGHTFORWARD
ASSIGN	RECODE
COPY	Datetime functions
CONCAT	STUDY_DY
BLANK	FUNCTION
WHERE	STACK

Straightforward functions



- Self-explanatory
- Simple
- Direct
- Easy

SOURCE_DS	SOURCE_VAR	TARGET_DS	TARGET_VAR	SPECIFICATION	FUNCTION
source.demog		DM	STUDYID	Assign value "EX001" for all records	ASSIGN[EX001]
source.demog	SUBJECT	DM	SUBJID	Copy from source variable	COPY
source.demog		DM	USUBJID	Concatenate STUDYID and SUBJID, separated by "-"	CONCAT[STUDYID||"-"||SUBJID]
source.demog		DM	DTHDTC	Set to missing	BLANK
source.vitals	OCCUR	VS		Select only the cases where vital signs were collected	WHERE[OCCUR = 1]

Recode

Metadata | Mapping | **Recode** | Order | Settings

```
if gender = "Male" then sex = "M";
else if gender = "Female" then sex = "F";
```

SOURCE_DS	SOURCE_VAR	TARGET_DS	TARGET_VAR	SPECIFICATION	FUNCTION
source.demog	DEMOG_SEX	DM	SEX	Recode according to the recode list GENDER	RECODE[GENDER]

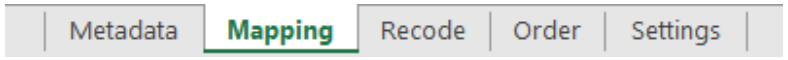
Metadata | **Mapping** | Recode | Order | Settings

CODELIST	TYPE	FROM	TO
GENDER	C2C	Male	M
GENDER	C2C	Female	F
GENDER	C2C		
NY	N2C	0	N
NY	N2C	1	Y
NY	N2C	.	

Metadata | Mapping | **Recode** | Order | Settings

The mapping engine also checks whether the source dataset contains values that are not in the recode list!

Datetime functions



- There are many functions to convert any sort of date/time (numeric, ISO, etc.) to any other type of date/time
  - Flexible and adaptable
  - Others can be created if and when needed

- STUDY_DY: Create any –DY variable based on a reference date/time variable

SOURCE_DS	SOURCE_VAR	TARGET_DS	TARGET_VAR	SPECIFICATION	FUNCTION
source.demog	DEMOG_DATE	DM	RFSTDTC	Create ISO datetime from the source variables DEMOG_DATE and DEMOG_TIME	ISODATETIME[ISO=DEMOG_DATE, TIME=DEMOG_TIME]
source.demog	DEMOG_TIME	DM	RFSTDTC	See derivation at DEMOG_DATE	
source.vitals	EXAM_DT	VS	VSDTC	Create ISO datetime from source ISO datetime	ISODATETIME[ISO=EXAM_DT]
		VS	VSDY	Derive VSDY as follows: VSDY = VSDTC-RFSTDTC+1 when VSDTC is on or after RFSTDTC. VSDY = VSDTC-RFSTDTC when VSDTC precedes RFSTDTC.	FUNCTION2[#STUDY_DY[REFDATE=RFSTDTC]#;]

Function

Metadata | **Mapping** | Recode | Order | Settings

- What if there isn't a specific engine-function available? Use the FUNCTION function!
  - Normal programming language

SOURCE_DS	SOURCE_VAR	TARGET_DS	TARGET_VAR	SPECIFICATION	FUNCTION
source.vitals	SUBJECT_VS	VS	USUBJID	Copy only the characters after the "."	FUNCTION[USUBJID=scan(SUBJECT_VS,2,".");]

STACK

Metadata | **Mapping** | Recode | Order | Settings

- Switch from a horizontal structure to a vertical structure → transpose data

SOURCE_DS	SOURCE_VAR	TARGET_DS	TARGET_VAR	SPECIFICATION	FUNCTION
source.vitals	HEIGHT	VS	VSTESTCD	For each record generate one "Height" record: Assign value "HEIGHT" for all records	STACK1[#ASSIGN[HEIGHT]#;]
source.vitals	HEIGHT	VS	VSORRES	For each record generate one "Height" record: Copy from source variable	STACK1[#COPY#;]
source.vitals	HEIGHTU	VS	VSORRESU	For each record generate one "Height" record: Copy from source variable when VSORRES is not missing	STACK1[if ^missing(VSORRES) then #COPY#; else #BLANK#;]
source.vitals	WEIGHT	VS	VSTESTCD	For each record generate one "Weight" record: Assign value "WEIGHT" for all records	STACK2[#ASSIGN[WEIGHT]#;]
source.vitals	WEIGHT	VS	VSORRES	For each record generate one "Weight" record: Copy from source variable	STACK2[#COPY#;]
source.vitals	WEIGHTU	VS	VSORRESU	For each record generate one "Weight" record: Copy from source variable when VSORRES is not missing	STACK2[if ^missing(VSORRES) then #COPY#; else #BLANK#;]

VISITNUM	HEIGHT	HEIGHTU	WEIGHT	WEIGHTU
1	158	cm		kg
2	191	cm	100	kg

VISITNUM	VSTESTCD	VSORRES	VSORRESU
1	HEIGHT	158	cm
2	HEIGHT	191	cm
1	WEIGHT		
2	WEIGHT	100	kg

STACK

Metadata | **Mapping** | Recode | Order | Settings

SOURCE_DS	SOURCE_VAR	TARGET_DS	TARGET_VAR	SPECIFICATION	FUNCTION
source.vitals	HEIGHT	VS	VSTESTCD	For each record generate one "Height" record: Assign value "HEIGHT" for all records	STACK1[#ASSIGN[HEIGHT]#;]
source.vitals	HEIGHT	VS	VSORRES	For each record generate one "Height" record: Copy from source variable	STACK1[#COPY#;]
source.vitals	HEIGHTU	VS	VSORRESU	For each record generate one "Height" record: Copy from source variable when VSORRES is not missing	STACK1[if ^missing(VSORRES) then #COPY#; else #BLANK#;]
source.vitals	WEIGHT	VS	VSTESTCD	For each record generate one "Weight" record: Assign value "WEIGHT" for all records	STACK2[#ASSIGN[WEIGHT]#;]
source.vitals	WEIGHT	VS	VSORRES	For each record generate one "Weight" record: Copy from source variable	STACK2[#COPY#;]
source.vitals	WEIGHTU	VS	VSORRESU	For each record generate one "Weight" record: Copy from source variable when VSORRES is not missing	STACK2[if ^missing(VSORRES) then #COPY#; else #BLANK#;]

STACK1-functions will result in:

```
VSTESTCD = "HEIGHT";
VSORRES= STRIP(PUT(HEIGHT, BEST.));
IF ^MISSING(VSORRES) THEN VSORRESU = STRIP(HEIGHTU);
  ELSE VSORRESU = "";
OUTPUT;
```

STACK2-functions will result in:

```
VSTESTCD = "WEIGHT";
VSORRES= STRIP(PUT(WEIGHT, BEST.));
IF ^MISSING(VSORRES) THEN VSORRESU = STRIP(WEIGHTU);
  ELSE VSORRESU = "";
OUTPUT;
```

Poststeps

Metadata | **Mapping** | Recode | Order | Settings

SOURCE_DS	SOURCE_VAR	TARGET_DS	TARGET_VAR	SPECIFICATION	FUNCTION
		VS		**POSTSTEP1: Combine source datasets.** Combine mapped_source_vitals with mapped_source_demog, mapped_source_vitals2	POSTSTEP1[data work.vs01; set work.mapped_source_vitals work.mapped_source_demog; run;]
		VS		**POSTSTEP2: Add SDTM DM.RFSTDTC to mapped VS data** Obtain RFSTDTC from OUTLIB.DM by merging the mapped VS data with OUTLIB.DM on USUBJID.	POSTSTEP2[proc sql; create table work.vs02 as select a.*, b.rfstdtc from work.vs01 as A left join outlib.dm as B on a.usubjid = b.usubjid; QUIT;]
		VS	VSDY	Derive VSDY as follows: VSDY = VSDTC-RFSTDTC+1 when VSDTC is on or after RFSTDTC. VSDY = VSDTC-RFSTDTC when VSDTC precedes RFSTDTC.	FUNCTION2[#STUDY_DY[REFDATE=RFSTDTC]#;]
		VS	VSBLFL	**POSTSTEP3: Add baseline flag**	POSTSTEP3[%add_blfl(inset=work.vs02 ,outset=work.vs03);]

Order

Metadata | Mapping | Recode | **Order** | Settings

- Set the creation order of the output datasets
- Important when one dataset depends on variables from another output dataset
- Order of source/target datasets in the Mapping tab does not matter

CREATION ORDER OF DOMAINS
DM
VS

Settings

Metadata | Mapping | Recode | Order | **Settings**

SETTING	EXPLANATION	VALUE
SOURCE	Define the path to the source library	C:\EXAMPLE\SOURCE
OUTLIB	Define the path for the output library	C:\EXAMPLE\OUTPUT
SUPP	Create supplemental qualifiers	Y
SHORT_LENGTHS	Shorten variable length to length of longest value	Y
XPT	Create XPT files	Y
KEEP_EMPTY	Output datasets with no records	N

Mapping Engine: Advantages

- Checks if:
  - All source variables are mapped
  - All source values are used with the recode list
  - No truncation occurs
  - Sorting on key variables gives unique records

- Reduces:
  - Repetitive code in one program
  - Repetitive coding for studies with same set-up

- Creation of a library of mapping standards
  - Re-using mapping for multiple studies, without changing a thing!

Conclusions

- The mapping engine makes data conversion faster and easier
- It is fully customisable to meet any needs and adapt to any situation
- Everything is organised and explained/commented in a single file
- A mapping library can be set up to be re-used
- It allows you to focus on what's important: data!

Thank you! Questions?

OCS Life Sciences

Sofia Vale: sofia.vale@ocs-consulting.com