Metadata-based Auto-Programming Process

Lisa Lyons, Janssen R&D US
Qian Zhao, Harry Chen, Janssen R&D China
Introduction
Traditionally the process for programming analysis datasets (e.g. ADaM) is cumbersome and heavily relies on manual work.

Regulatory agencies enforcing more standardized data and industry moving towards standardizing analysis requirements.

We need to achieve more automation for analysis by reusing standard and previous study algorithms stored in metadata to generate SAS code.

Illustrate the concept of metadata-based auto programming in ADaM dataset creation:

- **Macro Level** using ADaM planning specifications and Macro level metadata.
- **Data Step Level** using a structured metadata approach for IF-THEN/ELSE logic.
Example 1:

Macro Level Metadata
Macro Level Metadata

Example 1

- Automatically generate ADAE.sas
  - Input → ADaM planning sheet (specifications)
  - Process → Extracting macro call information from standard macro and program folders and convert to metadata

ADAE.xlsx Specification
Macro Level Metadata

Example 1 (cont’d)

- Automatically generate ADAEfmt.sas
  - Input → ADaM planning sheet (specifications)
  - Process → output format/informat from Codelist

Example 1 (cont’d)

<table>
<thead>
<tr>
<th>REFERENCE</th>
<th>TXAREA</th>
<th>DSAREA</th>
<th>CODELIST</th>
<th>DATATYPE</th>
<th>CODEVAL</th>
<th>RNK</th>
<th>DECOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADaM</td>
<td>Global</td>
<td>General</td>
<td>RELN</td>
<td>integer</td>
<td>1</td>
<td>1</td>
<td>NOT RELATED</td>
</tr>
<tr>
<td>ADaM</td>
<td>Global</td>
<td>General</td>
<td>RELN</td>
<td>integer</td>
<td>2</td>
<td>2</td>
<td>DOUBTFUL</td>
</tr>
<tr>
<td>ADaM</td>
<td>Global</td>
<td>General</td>
<td>RELN</td>
<td>integer</td>
<td>3</td>
<td>3</td>
<td>POSSIBLE</td>
</tr>
<tr>
<td>ADaM</td>
<td>Global</td>
<td>General</td>
<td>RELN</td>
<td>integer</td>
<td>4</td>
<td>4</td>
<td>PROBABLE</td>
</tr>
<tr>
<td>ADaM</td>
<td>Global</td>
<td>General</td>
<td>RELN</td>
<td>integer</td>
<td>5</td>
<td>5</td>
<td>VERY LIKELY</td>
</tr>
<tr>
<td>ADaM</td>
<td>Global</td>
<td>General</td>
<td>RELN</td>
<td>integer</td>
<td>6</td>
<td>6</td>
<td>MULTIPLE</td>
</tr>
<tr>
<td>ADaM</td>
<td>Global</td>
<td>General</td>
<td>RELGR</td>
<td>text</td>
<td>NOT RELATED</td>
<td>1</td>
<td>NOT RELATED</td>
</tr>
<tr>
<td>ADaM</td>
<td>Global</td>
<td>General</td>
<td>RELGR</td>
<td>text</td>
<td>RELATED</td>
<td>2</td>
<td>RELATED</td>
</tr>
<tr>
<td>ADaM</td>
<td>Global</td>
<td>General</td>
<td>RELGRN</td>
<td>integer</td>
<td>1</td>
<td>1</td>
<td>NOT RELATED</td>
</tr>
<tr>
<td>ADaM</td>
<td>Global</td>
<td>General</td>
<td>RELGRN</td>
<td>integer</td>
<td>2</td>
<td>2</td>
<td>RELATED</td>
</tr>
<tr>
<td>ADaM</td>
<td>Global</td>
<td>General</td>
<td>ADYES</td>
<td>text</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Macro Level Metadata

Example 1 (cont’d)

- `%uacADaMCode` macro call:

| Remarks                          | The ADMacrodir, studyADPrograms, and domain parameters can have multiple separated by a pipe (|). |
|----------------------------------|---------------------------------------------------------------------------------------------------|
|                                  | For example, &path1 | &path2 for ADMacrodir and studyADPrograms.                                                      |
|                                  | For example, ADAE | ADVS for domain.                                                                                |
| Macro Parameters                 | ADMacrodir – paths to reference macro folders                                                    |
|                                  | studyADPrograms – paths to reference program folders                                             |
|                                  | metadir – path to your metadata that will be used to create your code                            |
|                                  | domain – domains you want to create                                                               |
|                                  | outDir – output directory path write out files                                                   |

```
%uacADaMCode(ADMacrodir = &topdrv.\1_GLOBAL\AnalysisStandards_Test Release\Macros,
             ,metadir = &opath,
             ,studyADPrograms = &topdrv.\1_GLOBAL\AnalysisStandards_Test Release\Programs,
             ,domain = ADAE,
             ,outDir = &opath);
```


Macro Level Metadata

Example 1 (cont’d)

Extracting information (by Perl Regular Expression) from standard macro and program folders and converting it to macro level metadata

Macro Definition Metadata:

<table>
<thead>
<tr>
<th>macName</th>
<th>paramn</th>
<th>param</th>
<th>equal</th>
<th>paramval</th>
</tr>
</thead>
<tbody>
<tr>
<td>%MACRO ADREL</td>
<td>1</td>
<td>indsn</td>
<td>=</td>
<td></td>
</tr>
<tr>
<td>%MACRO ADREL</td>
<td>2</td>
<td>outdsn</td>
<td>=</td>
<td></td>
</tr>
<tr>
<td>%MACRO ADREL</td>
<td>3</td>
<td>keepvars</td>
<td>=</td>
<td></td>
</tr>
<tr>
<td>%MACRO ADREL</td>
<td>4</td>
<td>debug</td>
<td>=</td>
<td>OFF</td>
</tr>
<tr>
<td>%MACRO ADTRTEMFL</td>
<td>1</td>
<td>indsn</td>
<td>=</td>
<td></td>
</tr>
<tr>
<td>%MACRO ADTRTEMFL</td>
<td>2</td>
<td>invar</td>
<td>=</td>
<td></td>
</tr>
<tr>
<td>%MACRO ADTRTEMFL</td>
<td>3</td>
<td>adrule</td>
<td>=</td>
<td></td>
</tr>
<tr>
<td>%MACRO ADTRTEMFL</td>
<td>4</td>
<td>outdsn</td>
<td>=</td>
<td>OFF</td>
</tr>
</tbody>
</table>

Macro Call Metadata (Parameter value):

<table>
<thead>
<tr>
<th>domain</th>
<th>macName</th>
<th>paramn</th>
<th>param</th>
<th>equal</th>
<th>paramval</th>
</tr>
</thead>
<tbody>
<tr>
<td>adae</td>
<td>%ADREL</td>
<td>1</td>
<td>indsn</td>
<td>=</td>
<td>adae</td>
</tr>
<tr>
<td>adae</td>
<td>%ADREL</td>
<td>2</td>
<td>outdsn</td>
<td>=</td>
<td>adae</td>
</tr>
<tr>
<td>adae</td>
<td>%ADREL</td>
<td>3</td>
<td>keepvars</td>
<td>=</td>
<td>AERELN RELGR1 RELGR1N</td>
</tr>
<tr>
<td>adae</td>
<td>%ADREL</td>
<td>4</td>
<td>debug</td>
<td>=</td>
<td>OFF</td>
</tr>
<tr>
<td>adae</td>
<td>%ADTRTEMFL</td>
<td>1</td>
<td>indsn</td>
<td>=</td>
<td>adae</td>
</tr>
<tr>
<td>adae</td>
<td>%ADTRTEMFL</td>
<td>2</td>
<td>invar</td>
<td>=</td>
<td>&amp;dayvar</td>
</tr>
<tr>
<td>adae</td>
<td>%ADTRTEMFL</td>
<td>3</td>
<td>adrule</td>
<td>=</td>
<td>VERSION1</td>
</tr>
<tr>
<td>adae</td>
<td>%ADTRTEMFL</td>
<td>4</td>
<td>outdsn</td>
<td>=</td>
<td>adae</td>
</tr>
<tr>
<td>adae</td>
<td>%ADTRTEMFL</td>
<td>5</td>
<td>debug</td>
<td>=</td>
<td>OFF</td>
</tr>
</tbody>
</table>
Macro Level Metadata

Example 1 (cont’d)

**Autogenerated ADAE.sas:**

```sas
/* Short Description: Macro to create Treatment Emergent Analysis Flag 
* Input : input dataset indsn,d_in.ae 
* Output : input dataset in INDSN with TRTEMFL variable added to create the output dataset in OUTDSN */
%ADTRTEMFL(indsn = _09ADADURN, 
  invar=&dayvar, 
  adrule = VERSION1, 
  outdsn = _10ADTRTEMFL, 
  debug=OFF);

/* Short Description: Macro to create Causality & Pooled Causality Group variables 
* Input: input dataset is indsn,d_in.ae 
* Output: Input dataset in INDSN with AERELN, RELGR1, RELGR1N added to output dataset in OUTDSN */
%ADRELF(indsn = _10ADTRTEMFL, 
  outdsn = _11ADRELF, 
  keepvars=AERELN RELGR1 RELGR1N, 
  debug=OFF);
```

Auto-Created
Reference multiple previous studies and the standard library

Macro Level Metadata

Example 1 (cont’d)

Remarks
: The ADMacrodir, studyADPrograms, and domain parameters can
  have multiple separated by a pipe (|).
  For example, &path1 | &path2 for ADMacrodir and studyADPrograms.
  For example, ADAE | ADVS for domain.

Macro Parameters
: ADMacrodir - paths to reference macro folders
: studyADPrograms - paths to reference program folders
: metadir - path to your metadata that will be used to create your code
: domain - domains you want to create
: outDir - output directory path write out files

$uacADaMCode{ADMacrodir = &study1MacroPath | &study2MacroPath | &study32MacroPath | &standardMacroPath
 ,metadir = &opath
 ,studyADPrograms = &study1ProgramPath | &study2ProgramPath | &study3ProgramPath | &standardProgramPath
 ,domain = ADSL | ADAE | ADVS | ADLB
 ,outDir = &opath);
**%PowerMacroExplorer**

- **Features**
  - Compare macro and program metadata from multiple studies to obtain a complete overview
  - Easy to find the most commonly used macros and the study-specific macros
  - Learn from the metadata and pick the best one for their own needs to apply in the `%uacADaMCode` macro

**Example 1 (cont’d) Macro Level Metadata**
Benefits

- Using the standards and previous studies reference save a lot time.
- Macros have already been tested and validated
- Don’t need to ‘re-invent the wheel’
- Keep the code more consistent
- Allows for macro governance across a disease area or therapeutic areas, ensure robust macro development and consistency
Example 2:

IF-THEN/ELSE logic in Data Step Level
Example 2

- Assume we have the following data step specification for IF-THEN/ELSE logic

- We can call `%ifstatement` macro against this specification to generate the SAS program

- The above macro call generates these IF-THEN/ELSE program logic
## IF-THEN/ELSE logic in Data Step Level

### Example 2 (cont’d)

Table 1 - Standardized *ifVar_outVar* metadata structure used to autogenerate complex IF-THEN/ELSE logic

<table>
<thead>
<tr>
<th>groupVar</th>
<th>ifVar1</th>
<th>ifVar2</th>
<th>...</th>
<th>ifVarN</th>
<th>outVar1</th>
<th>outVar2</th>
<th>...</th>
<th>outVarN</th>
</tr>
</thead>
<tbody>
<tr>
<td>xxx</td>
<td>xxx</td>
<td>xxx</td>
<td>xxx</td>
<td>xxx</td>
<td>xxx</td>
<td>xxx</td>
<td>xxx</td>
<td>xxx</td>
</tr>
<tr>
<td>xxx</td>
<td>xxx</td>
<td>xxx</td>
<td>xxx</td>
<td>xxx</td>
<td>xxx</td>
<td>xxx</td>
<td>xxx</td>
<td>xxx</td>
</tr>
<tr>
<td>xxx</td>
<td>xxx</td>
<td>xxx</td>
<td>xxx</td>
<td>xxx</td>
<td>xxx</td>
<td>xxx</td>
<td>xxx</td>
<td>xxx</td>
</tr>
</tbody>
</table>

Table 2 - *ifVar_outVar* metadata values populated

<table>
<thead>
<tr>
<th>groupVar</th>
<th>ifVar1</th>
<th>ifVar2</th>
<th>outVar1</th>
<th>outVar2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>varA</td>
<td>‘valueA’</td>
<td>‘valueX’</td>
<td>‘valueY’</td>
</tr>
<tr>
<td>1</td>
<td>varA</td>
<td>‘valueB’</td>
<td>varX</td>
<td>varY</td>
</tr>
<tr>
<td>2</td>
<td>#var1&gt;var2</td>
<td>‘valueC’</td>
<td>#expressionX</td>
<td>#expressionY</td>
</tr>
</tbody>
</table>

### %ifStatement macro call

```sql
if ifVar1=varA then do;
    if ifVar2='valueA' then do; outVar1='valueX'; outVar2='valueY'; end;
    else if ifVar2='valueB' then do; outVar1=varX; outVar2=varY; end;
end;
if var1>var2 and ifVar2='valueC' then do; expressionX; expressionY; end;
```
Lab Toxicity Grade Derivation

Example 2 (cont’d)

IF-THEN/ELSE logic in Data Step Level

**Auto-Created**

Intermediate Data

%ifStatement macro call
### Best Overall Response (RECIST criteria)

<table>
<thead>
<tr>
<th>Overall response at 1st time point (X₁)</th>
<th>Overall response at 2nd time point (X₂)</th>
<th>Overall response at 3rd time point (X₃)</th>
<th>Best overall Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>CR or PR</td>
<td>CR or PR</td>
<td>CR or PR</td>
<td>CR or PR</td>
</tr>
<tr>
<td>CR or PR</td>
<td>CR or PR</td>
<td>CR or PR</td>
<td>CR or PR</td>
</tr>
<tr>
<td>CR or PR</td>
<td>CR or PR</td>
<td>CR or PR</td>
<td>CR or PR</td>
</tr>
<tr>
<td>CR or PR</td>
<td>CR or PR</td>
<td>CR or PR</td>
<td>CR or PR</td>
</tr>
</tbody>
</table>

**IF-THEN/ELSE logic in Data Step Level**

```
if R1='CR' then do;
  if MISSING(R2) and MISSING(R3) then BOR=IFC(X1 > XX,"SU","NE");
  else if R2='CR' then do;
    if X₂ - X₁ > YY then BOR='CR';
    else if R₂='PR' then do;
      NOTE='Note: cannot be PR after CR.';
      BOR=IFC(X1 > XX,"SP","PD");end;
    else if R₂='SD' then do;
      NOTE='Note: cannot be SD after CR.';
      BOR=IFC(X1 > XX,"SP","PD");end;
    else if R₂='PD' then do;
      BOR=IFC(X1 > XX,"SU","PD");end;
    else do: put 'ERROR: Unexpected error 1.' usu8601=0;end;
  end;
  else if R₂='PR' then do:
    NOTE='Note: cannot be PR after CR.';
    BOR=IFC(X1 > XX, "SU", "PD");end;
  else if R₂='SD' then do:
    NOTE='Note: cannot be SD after CR.';
    BOR=IFC(X1 > XX, "SU", "PD");end;
  else if R₂='PD' then do:
    BOR=IFC(X1 > XX, "SU", "PD");end;
  else do: put 'ERROR: Unexpected error 2.' usu8601=0;end;
end;
else if R₁='PR' then do:
  NOTE='Note: cannot be PR after CR.';
  BOR=IFC(X1 > XX, "SU", "PD");end;
else if R₁='SD' then do:
  NOTE='Note: cannot be SD after CR.';
  BOR=IFC(X1 > XX, "SU", "PD");end;
else if R₁='PD' then do:
  BOR=IFC(X1 > XX, "SU", "PD");end;
else do: put 'ERROR: Unexpected error 3.' usu8601=0;end;
```

---

### Example 2 (cont’d)

- **IF** - THEN/ELSE logic in Data Step Level
- **Auto-Created**
- **%ifStatement**
In the previous examples, the macro `%ifStatement` serves as a compiler of the ifVar_outVar structure metadata specification.

For each cell in the metadata specification, the macro distinguishes its function (value, variable, or expression, ifvar, outvar), and translates the metadata specification into the programming language.

IF-THEN/ELSE logic is a powerful, logical construct that can cover many data derivations that may or may not be achieved using other SAS techniques. We can specify value, variable and even expressions in the metadata table.
Summary

- We should try to achieve as much automation as possible, although 100% of automation is not yet possible.

- Benefits
  - Reduce manual work, minimize human error which often leads to wasted time in both coding and debugging
  - Programmer focuses on design, complex analysis rather than tedious and manual coding
  - Able to compare with standard and historical algorithms by metadata comparison to enhance the standards and maintain consistency
  - Centralized control of coding instead of scattered pieces
  - Potentially require lower level of SAS skills for more routine coding
  - Further benefits with future database and interface like MDR (MetaData Repository)
  - Concept can be extended to more complicated cases
Acknowledgements

We would like to thank the Janssen Autocode team Nick Masel, David Fan, Ilse Augustyns, Phyllis Wolf, Sumesh Kalappurakal, Taku Uryu, James Li, Sendil Ramabadran and Sandy Lei, for the contribution to the macros described in this paper and valuable comments.
Metadata-based Auto-Programming Process

Questions?

Lisa Lyons
Janssen R&D US
Phone: (609)-802-6283
e-mail: llyons3@its.jnj.com

Qian Zhao
Janssen R&D China.
Phone: 86-21-3337-7624
e-mail: qzhao1@its.jnj.com

Harry Chen
Janssen R&D China.
Phone: 86-21-3337-8892
e-mail: hchen104@its.jnj.com
Backup Slides
IF-THEN/ELSE logic in Data Step Level

Example 2 (cont’d)

Best Overall Response (RECIST criteria)

<table>
<thead>
<tr>
<th>Overall response at 1st time point (X₁)</th>
<th>Overall response at 2nd time point (X₂)</th>
<th>Overall response at 3rd time point (X₃)</th>
<th>Best overall response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unknown</td>
<td>Unknown</td>
<td>Unknown</td>
<td>Final: SD if X₁ = XX otherwise, NE Interim: if no additional disease assessments are expected then SD if X₂ = XX otherwise, NE otherwise uCR</td>
</tr>
<tr>
<td>CR with X₁ = YY</td>
<td>Any</td>
<td>Unknown</td>
<td>Final: SD if X₁ = XX otherwise, NE Interim: if no additional disease assessments are expected then SD if X₂ = XX otherwise, NE otherwise uCR</td>
</tr>
<tr>
<td>CR with X₁ = -YY</td>
<td>Any</td>
<td>Unknown</td>
<td>Final: SD if X₁ = XX otherwise, NE Interim: if no additional disease assessments are expected then SD if X₂ = XX otherwise, NE otherwise uCR</td>
</tr>
<tr>
<td>PR</td>
<td>Flag as a data issue with note; cannot be PR after CR; SD if X₁ = XX otherwise, PD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SD</td>
<td>Flag as a data issue with note; cannot be SD after CR; SD if X₁ = XX otherwise, PD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PD</td>
<td>SD if X₁ = XX otherwise, PD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PR</td>
<td>Any</td>
<td>Any</td>
<td>Final: SD if X₁ = XX otherwise, NE Interim: if no additional disease assessments are expected then SD if X₂ = XX otherwise, NE otherwise uCR</td>
</tr>
<tr>
<td>SD</td>
<td>Any</td>
<td>Any</td>
<td>Final: SD if X₁ = XX otherwise, NE Interim: if no additional disease assessments are expected then SD if X₂ = XX otherwise, NE otherwise uCR</td>
</tr>
<tr>
<td>PD</td>
<td>Any</td>
<td>Any</td>
<td>Final: SD if X₁ = XX otherwise, NE Interim: if no additional disease assessments are expected then SD if X₂ = XX otherwise, NE otherwise uCR</td>
</tr>
</tbody>
</table>

Code to convert spec to the ifVar_outVar metadata structure for use with %ifStatement macro call

```plaintext
%ifStatement
```

Auto-Created

Intermediate Data

```plaintext
if R1='CR' then do;
  if MISSING(R2) and MISSING(R3) then BOR=IFC(X1 > XX,"SU","NE");
else if R2='CR' then do;
  if X1 - X1 = YY then BOR="CR";
  else if X1 - X1 < YY then do;
    if MISSING(R3) then BOR=IFC(X2 > XX,"SU","NE");
  else if R3='PR' then do; NOTE=Note: cannot be PR after CR.; BOR=IFC(X1 > XX,"SD","PD");end;
  else if R3='SD' then do; NOTE=Note: cannot be SD after CR.; BOR=IFC(X1 > XX,"SD","PD");end;
  else if R3='PD' then do BOR=IFC(X2 > XX,"PD","PD");end;
  else do; put 'ERROR: Unexpected error 1.' USUBID= RSUIDC='BOR='';end;
end;
else if R2='PR' then do; NOTE=Note: cannot be PR after CR.; BOR=IFC(X1 > XX,"SD","PD");end;
else if R2='SD' then do; NOTE=Note: cannot be SD after CR.; BOR=IFC(X1 > XX,"SD","PD");end;
else if R2='PD' then do BOR=IFC(X2 > XX,"PD","PD");end;
else do; put 'ERROR: Unexpected error 1.' USUBID= RSUIDC='BOR='';end;
end;
else if R3='PR' then do;
  if MISSING(R2) and MISSING(R3) then BOR=IFC(X1 > XX,"SU","NE");
else if R3='CR' then do;
  if R3='CR' then do:
    if X1 - X2 >Y then BOR="CR";
  else do; put 'ERROR: Unexpected error 3.' USUBID= RSUIDC='BOR='';end;
end;
end;
end;
end;
```
Example 2

Benefits

- IF-THEN/ELSE logic is a powerful, logical construct that can cover many data derivations that may or may not be achieved using other SAS techniques. We can specify value, variable and even expressions in the metadata table.

- Maintain consistency between the specification and the program; therefore, changes need to be made only in one place. The code generated is easy to review and understand.

- Allow making changes in a central location and obtain better traceability from the source variable to the mapped variable.

- Enable a higher-level standardization due to reusability of the mapping specification template and the macro.

- The metadata may be saved in a general structured database to store most of the variable mapping algorithms to build a library