

Galápagos

Wild Merge

Automated Data Reconciliation in Both SAS and R

PHUSE EU Connect 2021

15th – 19th November 2021

Gerard Hermus

Machiel Visser





Disclaimer

- The views or opinions shared by the presenters do not necessarily reflect the policy or position of Galapagos
- The methods and code are shared for educational purposes only and are not meant to be taken as validated solutions



Preface 1

- Any programmer will have faced challenges when merging different sources with overlapping information:
 - Perhaps to just merge for SDTM creation
 - Record mismatches
 - Data mismatches
 - Cartesian products
 - Or reconciliation, which has the above challenges, plus:
 - A need to optimally use the data available to squeeze clues out to generate queries
- We realized that most of our manual methods followed a few rules which could be automated



Preface 2

- Our programming team contains both SAS and R programmers
- An unfriendly competition was started to solve the conundrum
- The final solution is an amalgamation of the efforts

Please note that the remainder of the presentation assumes that efforts have been made to align the sources in terms of variable names / content.



Wild merge overview

1. Determining key variables
2. Possible combinations of key variables
3. Determine importance / order of key variable combinations
4. Iterative merges
5. Combine merges into final merge
6. List differences between datasets for reconciliation



Wild merge algorithm

Input datasets

- Let's take the following two datasets to see how the wild merge algorithm works

CRF

USUBJID	XXDTC	XXREFID
1	01-Jan-20	1123
1	06-Jan-20	1124
1	07-Jan-20	1125
1	13-Jan-20	1127
2	17-Jan-20	1127
2	17-Jan-20	1128
2	25-Jan-20	1129
2	29-Jan-20	1130
3	29-Jan-20	1131

EXT

USUBJID	XXDTC	XXREFID	XXORRES
1	01-Jan-20	1123	4
1	05-Jan-20	1124	7.6
1	05-Jan-20	1124	5.1
1	09-Jan-20	1125	3.2
1	13-Jan-20	1126	8.6
2	17-Jan-20	1127	2.8
2	21-Jan-20	1128	2.1
2	25-Jan-20	1129	2.6
2	29-Jan-20	1130	2
4	29-Jan-20	1131	0.5



Wild merge algorithm

Determining key variables

- Automatically determine key variables to merge on:

- Retrieve metadata of both datasets
- Determine variables in common to build key

CRF	USUBJID	SITEID	XXDTC	XXREFID	
EXT	USUBJID		XXDTC	XXREFID	XXORRES

- By default, the system tags USUBJID as an important key variable which can't be dropped
 - This avoids merges that make no sense
 - Possible to adapt list of variables that should be included or excluded
 - E.g. for lab data with vial barcodes



Wild merge algorithm

Combinations of key variables

- Possible key combinations with all 3 variables:

USUBJID XXDTC XXREFID
USUBJID XXDTC
USUBJID XXREFID
XXDTC XXREFID
XXDTC
XXREFID
USUBJID

- Note how quickly the number of possible combinations goes up..

- Possible key combinations when USUBJID must be included:

USUBJID XXDTC XXREFID
USUBJID XXDTC
USUBJID XXREFID
USUBJID



Wild merge algorithm

Sidetrack: too many combinations of key variables, how to prevent a server crash?

- Example: two datasets have 27 variables in common:

- Total number of variable combinations =
$$\sum_{r=1}^n \frac{n!}{r! (n-r)!}$$

n = total number of common variables

r = number of variables included in key

- Number of variables in the key ranging from 1 to 27:

- $27! / (1! * (27 - 1)!) + 27! / (2! * (27 - 2)!) + \dots + 27! / (27! * (27 - 27)!) + 1$

- = 27 + 351 + ... + 1

- = 134,217,727 possible keys to merge on!

- Stop execution if > 200 combinations

- Limit can be changed



Wild merge algorithm

Order of merges

- Multiple merges possible based on different keys. But in which order?
- Determine for each key:
 - Explanatory power: number of unique records for key variables in both datasets
 - Number of variables in key

perm_key_sas	unique_rec	mergen
USUBJID XXDTC XXREFID	14	1
USUBJID XXDTC	12	2
USUBJID XXREFID	11	3
USUBJID		4

- To maximize precision, base order of iterative merges on:
 - Largest explanatory power of key
 - Largest number of variables in key
 - Alphabetical order of key



Wild merge algorithm

Indepth 1: iterative merges with USUBJID

perm_key_sas	unique_rec	mergen
USUBJID XXDTC XXREFID	14	1
USUBJID XXDTC	12	2
USUBJID XXREFID	11	3
USUBJID		4

CRF			EXT		
USUBJID	XXDTC	XXREFID	USUBJID	XXDTC	XXREFID
1	01-Jan-20	1123	1	01-Jan-20	1123
1		1124	1		1124
1		1125	1		1124
1	13-Jan-20		1		1125
2	17-Jan-20	1127	1	13-Jan-20	
2		1128	2	17-Jan-20	1127
2	25-Jan-20	1129	2		1128
2	29-Jan-20	1130	2	25-Jan-20	1129
3	29-Jan-20	1131	2	29-Jan-20	1130
			4	29-Jan-20	1131

1. Merge on **USUBJID XXDTC XXREFID**
 2. Remove successfully merged records
 3. Merge on **USUBJID XXDTC**
 4. Remove successfully merged records
 5. Merge on **USUBJID XXREFID**
 6. Remove successfully merged records
 7. Not possible to merge on **USUBJID**
- Two unmatched records left
 - Allow keys without USUBJID



Wild merge algorithm

Indepth 2: iterative merges without USUBJID as requirement

perm_key_sas	unique_rec	mergen
USUBJID XXDTC XXREFID	14	1
XXDTC XXREFID	13	2
USUBJID XXDTC	12	3
USUBJID XXREFID	11	4
XXDTC	10	5
XXREFID	9	6
USUBJID	4	7

CRF			EXT		
USUBJID	XXDTC	XXREFID	USUBJID	XXDTC	XXREFID
1	01-Jan-20	1123	1	01-Jan-20	1123
1		1124	1		1124
1		1125	1		1124
1	13-Jan-20		1		1125
2	17-Jan-20	1127	1	13-Jan-20	
2		1128	2	17-Jan-20	1127
2	25-Jan-20	1129	2		1128
2	29-Jan-20	1130	2	25-Jan-20	1129
	29-Jan-20	1131	2	29-Jan-20	1130
				29-Jan-20	1131

1. Merge on **USUBJID XXDTC XXREFID**
2. Remove successfully merged records
- ➔ 3. Merge on **XXDTC XXREFID**
4. Remove successfully merged records
5. Merge on **USUBJID XXDTC**
6. Remove successfully merged records
7. Merge on **USUBJID XXREFID**
8. All records have been merged



Wild merge algorithm

Indepth 3: duplicates

- How to deal with duplicates?
 - Duplicates from the CRF dataset are not accepted
 - Duplicates coming from the “second” dataset are accepted

CRF			EXT				
USUBJID	XXDTC	XXREFID	USUBJID	XXDTC	XXREFID	XXTESTCD	XXORRES
1	01-Jan-20	1123	1	01-Jan-20	1123	TEST1	4
1	06-Jan-20	1124	1	05-Jan-20	1124	TEST1	7.6
1	07-Jan-20	1125	1	05-Jan-20	1124	TEST2	5.1
1	13-Jan-20	1127	1	09-Jan-20	1125	TEST1	3.2
2	17-Jan-20	1127	1	13-Jan-20	1126	TEST1	8.6
2	17-Jan-20	1128	2	17-Jan-20	1127	TEST1	2.8
2	25-Jan-20	1129	2	21-Jan-20	1128	TEST1	2.1
2	29-Jan-20	1130	2	25-Jan-20	1129	TEST1	2.6
3	29-Jan-20	1131	2	29-Jan-20	1130	TEST1	2
			4	29-Jan-20	1131	TEST1	0.5

- Allows macro to be used for other activities:
 - Still manageable in terms of reconciliation output
 - Merging CRF LB sample dates with LB results



Wild merge algorithm

Output: reconciliation

- With all data merged we can analyze the data for differences

⊕ ds1_USUBJID	⊕ ds2_USUBJID	⊕ ds1_XXDTC	⊕ ds2_XXDTC	⊕ ds1_XXREFID	⊕ ds2_XXREFID
1	1	01JAN2020	01JAN2020	1123	1123
1	1	06JAN2020	05JAN2020	1124	1124
1	1	06JAN2020	05JAN2020	1124	1124
1	1	07JAN2020	09JAN2020	1125	1125
1	1	13JAN2020	13JAN2020	1127	1126
2	2	17JAN2020	17JAN2020	1127	1127
2	2	17JAN2020	21JAN2020	1128	1128
2	2	25JAN2020	25JAN2020	1129	1129
2	2	29JAN2020	29JAN2020	1130	1130
3		29JAN2020		1131	
	4		29JAN2020		1131



Wild merge algorithm

Output: reconciliation

perm_key_sas	unique_rec	mergen
USUBJID XXDTC XXREFID	14	1
USUBJID XXDTC	12	2
USUBJID XXREFID	11	3

CRF			EXT		
USUBJID	XXDTC	XXREFID	USUBJID	XXDTC	XXREFID
1	01-Jan-20	1123	1	01-Jan-20	1123
1		1124	1		1124
1		1125	1		1124
1	13-Jan-20		1		1125
2	17-Jan-20	1127	1	13-Jan-20	
2		1128	2	17-Jan-20	1127
2	25-Jan-20	1129	2		1128
2	29-Jan-20	1130	2	25-Jan-20	1129
3	29-Jan-20	1131	2	29-Jan-20	1130
			4	29-Jan-20	1131

- If all key variables match: no checks
- If USUBJID XXDTC match: show the difference in XXREFID
- If USUBJID XXREFID match: show the difference in XXDTC
- If no key variables match: tag records as such



Wild merge algorithm

Output: reconciliation

- Checks from call requiring USUBJID in every key

check	USUBJID	XXDTC	XXREFID
	1	01JAN2020	1123
Record was in both datasets, but DS2 was not unique XXDTC is unequal	1	DS1: 06JAN2020 DS2: 05JAN2020	1124
XXDTC is unequal	1	DS1: 07JAN2020 DS2: 09JAN2020	1125
XXREFID is unequal	1	13JAN2020	DS1: 1127 DS2: 1126
	2	17JAN2020	1127
XXDTC is unequal	2	DS1: 17JAN2020 DS2: 21JAN2020	1128
	2	25JAN2020	1129
	2	29JAN2020	1130
Record is only in DS1	3	29JAN2020	1131
Record is only in DS2	4	29JAN2020	1131



Wild merge algorithm

Output: reconciliation

- Checks from call **without** requiring USUBJID in keys

check	USUBJID	XXDTC	XXREFID
	1	01JAN2020	1123
Record was in both datasets, but DS2 was not unique XXDTC is unequal	1	DS1: 06JAN2020 DS2: 05JAN2020	1124
XXDTC is unequal	1	DS1: 07JAN2020 DS2: 09JAN2020	1125
XXREFID is unequal	1	13JAN2020	DS1: 1127 DS2: 1126
	2	17JAN2020	1127
XXDTC is unequal	2	DS1: 17JAN2020 DS2: 21JAN2020	1128
	2	25JAN2020	1129
	2	29JAN2020	1130
USUBJID is unequal	DS1: 3 DS2: 4	29JAN2020	1131



Wild merge algorithm

Output: final merge

- When having ongoing imperfect data but need to produce outputs regardless
 - Decisions are made mechanically with known rules
 - Creates reconciliation outputs for data management

△ CHECK	△ USUBJID	△ XXDTC	△ XXREFID	△ SITEID	⊕ XXORRES
	1	01JAN2020	1123	site1	4
Record was in both datasets, ...	1	06JAN2020	1124	site1	5.1
Record was in both datasets, ...	1	06JAN2020	1124	site1	7.6
XXDTC is unequal	1	07JAN2020	1125	site1	3.2
XXREFID is unequal	1	13JAN2020	1127	site1	8.6
	2	17JAN2020	1127	site2	2.8
XXDTC is unequal	2	17JAN2020	1128	site2	2.1
	2	25JAN2020	1129	site2	2.6
	2	29JAN2020	1130	site2	2
Record is only in DS1	3	29JAN2020	1131	site1	
Record is only in DS2	4	29JAN2020	1131		0.5



Wild merge

Programs

- Implemented in both SAS and R
- Code is available on GitHub
 - <https://github.com/vissermachi/WildMerge>
- Questions about the wild merge code:
 - Gerard.Hermus@glpg.com
 - Machiel.Visser@glpg.com