



siera: Redefining TFL Automation

Anna Yaggi and Shivani Gupta

October 15th, 2025

Author Bios



Anna Yaggi
Product Manager



Malan Bosman
Lead Automation
Engineer



Shivani Gupta
Director of Statistical
Programming

CDISC Analysis Results Standard - a Logical Data Model



Analysis Results Standard (ARS) v1.0



Large trials generate many analysis results in the form of tables, figures, and written reports, yet these results are rarely output in a form that is machine-readable. Previously, there has been no standard way of describing and organizing these results, making it difficult to automate their generation, make them reproducible, trace their origin, or enable them to be reused in other outputs.

To address these inefficiencies, CDISC has developed the [Analysis Results Standard \(ARS\)](#), which aim to facilitate automation, reproducibility, reusability, and traceability of analysis results data.

Features of ARS v1.0

- A Logical Data Model that describes analysis results and associated metadata.
- A User Guide to illustrate and exercise the model with common safety displays.

<https://cdisc-org.github.io/analysis-results-standard/>

Analysis Results Standard (ARS)

Search

Analysis Results Standard (ARS)

Schema Diagram

Classes

Slots

Enumerations

Types

Subsets

Analysis Results Standard (ARS)

Logical model to support both the prospective specification of analyses and the fully contextualized representation of the results of the analyses.

URI: <https://www.cdisc.org/ars/1-0> Name: ars_ldm

Schema Diagram

Classes

Classes provide templates for organizing data. Data objects instantiate slots (aka fields, attributes) that are applicable to it. See [LinkML docs](#).

Class	Description
NamedObject	An object with a name
ReportingEvent	A set of analyses and o requirements...
ListOfContents	A structured list of anal

cdisc

Analysis Results Standard User Guide

Version 1.0 (Final)

Prepared by the
Analysis Results Standard Team

Notes to Readers

- This is the final Version 1.0 of the Analysis Results Standard User Guide.
- This document is based on ADaM v2.1 and Analysis Results Metadata (ARM) v1.0 for Define-XML v2.0

Revision History

Date	Version
2024-04-19	Final

© 2024 Clinical Data Interchange Standards Consortium, Inc. All rights reserved.

<https://wiki.cdisc.org/display/ARSP/Analysis+Results+Standard+User+Guide+v1.0>

Model Components

Analysis Set

Data Subset

Analysis Grouping

Analysis Method

Analysis

Results

Summary of Demographics

Study - CDISC 360

Page x of y

Table 14.1.1
Summary of Demographics
Safety Population

Characteristics	Placebo (N=XX)	Xanomeline Low Dose (N=XX)	Xanomeline High Dose (N=XX)
Age (years)			
n	XX	XX	XX
Mean (SD)	XX.X (XX.XX)	XX.X (XX.XX)	XX.X (XX.XX)
Median	XX.X	XX.X	XX.X
Q1, Q3	XX.X, XX.X	XX.X, XX.X	XX.X, XX.X
Min, Max	XX, XX	XX, XX	XX, XX
Age Group, n (%)			
< 65 years	XX (XX.X)	XX (XX.X)	XX (XX.X)
≥ 65 years	XX (XX.X)	XX (XX.X)	XX (XX.X)
Gender, n (%)			
Male	XX (XX.X)	XX (XX.X)	XX (XX.X)
Female	XX (XX.X)	XX (XX.X)	XX (XX.X)
Ethnicity, n (%)			
Hispanic or Latino	XX (XX.X)	XX (XX.X)	XX (XX.X)
Not Hispanic or Latino	XX (XX.X)	XX (XX.X)	XX (XX.X)

Output

Source dataset: adsl, Generated on: DDMONYYYY:HH:MM

Program: <pid>.sas, Output: <pid><oid>.rtf, Generated on: DDMONYYYY:HH:MM

Output

Source dataset: adsl, Generated on: DDMONYYYY:HH:MM
Program: <pid>.sas, Output: <pid><oid>.rtf, Generated on: DDMONYYYY:HH:MM

Summary of TEAE by SOC and PT

Study - CDISC 360

Page x of y

Table 14.3.1.1

Summary of TEAE by System Organ Class and Preferred Term

Safety Population

System Organ Class	Placebo	Xanomeline Low Dose	Xanomeline High Dose
Preferred Term [a], n (%)	(N=XX)	(N=XX)	(N=XX)
Number of subjects with at least one event	XX (XX.X)	XX (XX.X)	XX (XX.X)
<SOC 1>	XX (XX.X)	XX (XX.X)	XX (XX.X)
<Preferred Term 1>	XX (XX.X)	XX (XX.X)	XX (XX.X)
...	XX (XX.X)	XX (XX.X)	XX (XX.X)
<Preferred Term n>	XX (XX.X)	XX (XX.X)	XX (XX.X)
<SOC 2>	XX (XX.X)	XX (XX.X)	XX (XX.X)
<Preferred Term 1>	XX (XX.X)	XX (XX.X)	XX (XX.X)
...	XX (XX.X)	XX (XX.X)	XX (XX.X)
<Preferred Term n>	XX (XX.X)	XX (XX.X)	XX (XX.X)

Notes: TEAE=Treatment-Emergent Adverse Events.
Subjects are counted once within each system organ class and preferred term.
[a] All investigators adverse events were coded using MedDRA version xx.x.

Source dataset: adae, Generated on: DDMMYYYY:HH:MM
Program: <pid>.sas, Output: <pid><oid>.rtf, Generated on: DDMMYYYY:HH:MM

Notes: TEAE=Treatment-Emergent Adverse Events.
Subjects are counted once within each system organ class and preferred term.
[a] All investigators adverse events were coded using MedDRA version xx.x.

Source dataset: adae, Generated on: DDMONYYYY:HH:MM
Program: <pid>.sas, Output: <pid><oid>.rtf, Generated on: DDMONYYYY:HH:MM

Source: CDISC

■ Download via TFL Designer Community Version (tfl designer.org)

- You can build shells and download their full ARS metadata using the community edition of TFL Designer



■ Machine-Readable JSON Format

- CDISC ARS metadata is provided in a structured, machine-readable JSON format.
- This allows for efficient integration and downstream processing.

■ Easily Visualized in Excel

DataSubsets

AnalysisSets

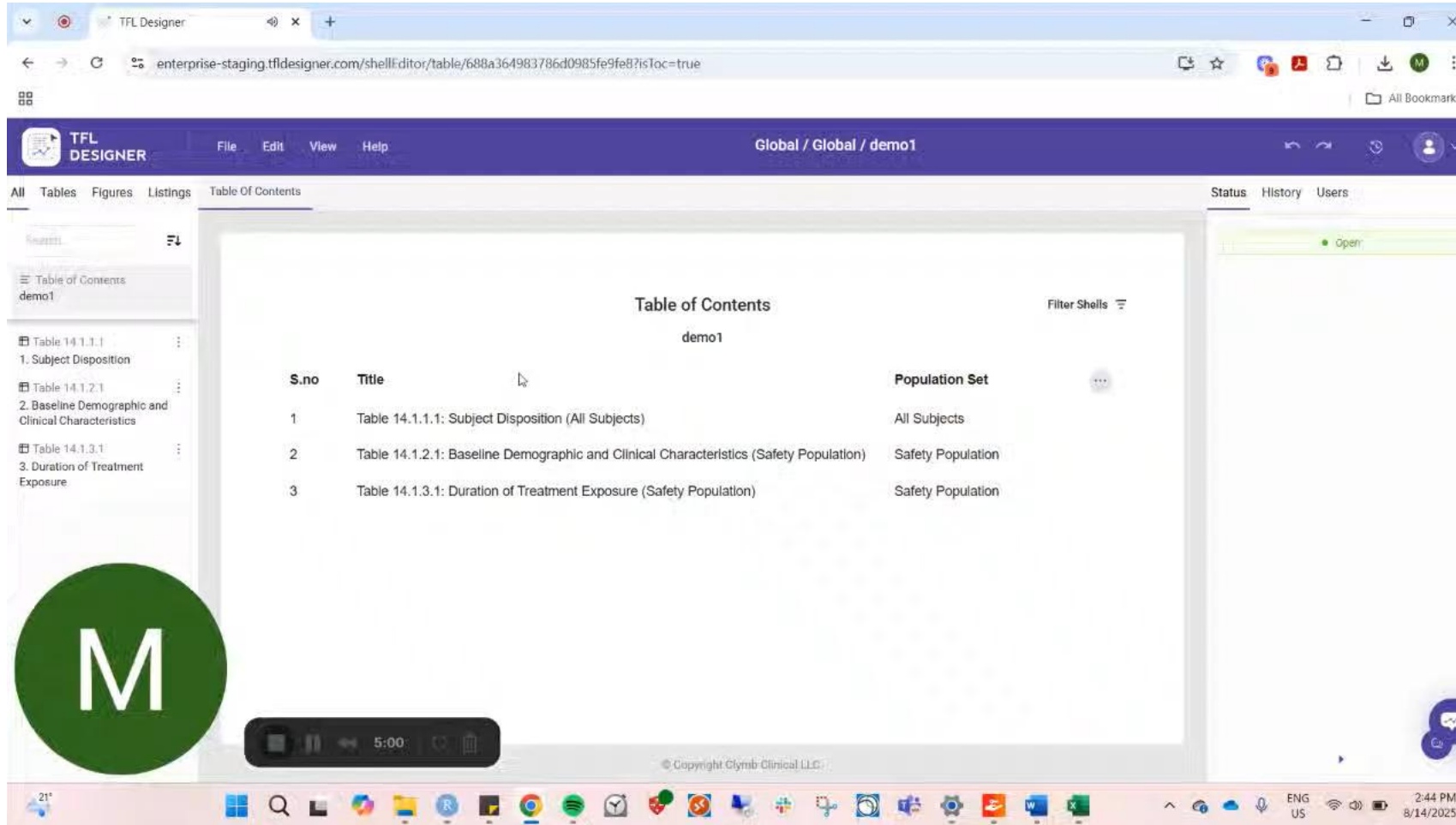
AnalysisGroupings

Analyses

AnalysisResults

- The JSON can be converted into an Excel workbook for human-friendly inspection.
- Each ARS class appears as a separate tab in the spreadsheet.

Extracting Metadata from TFL Designer



The screenshot shows the TFL Designer web application interface. The browser address bar displays the URL: `enterprise-staging.tfldesigner.com/shellEditor/table/688a364983786d0985fe9fe8?isToc=true`. The application header includes the TFL Designer logo and navigation tabs: File, Edit, View, and Help. The main content area is titled "Table of Contents" and displays a table with the following data:

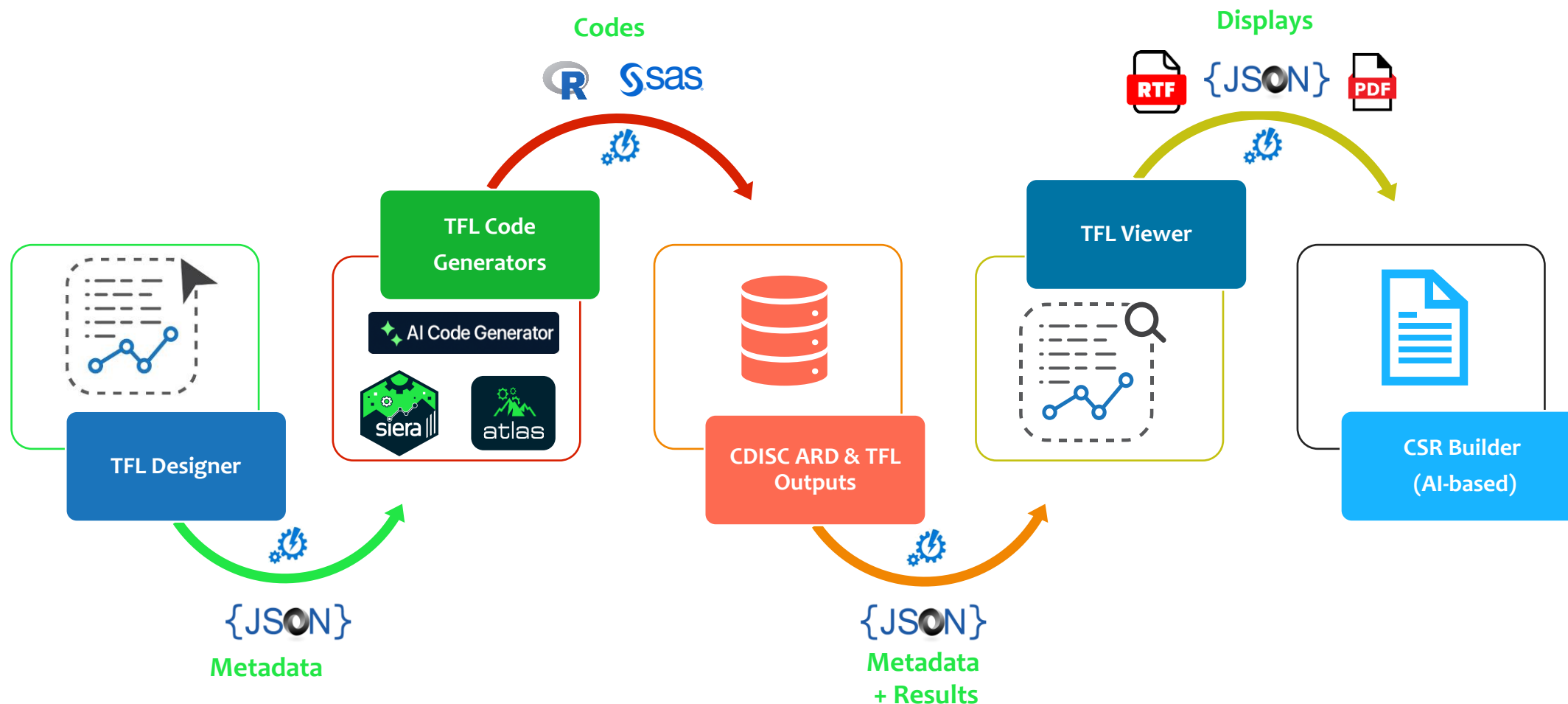
S.no	Title	Population Set
1	Table 14.1.1.1: Subject Disposition (All Subjects)	All Subjects
2	Table 14.1.2.1: Baseline Demographic and Clinical Characteristics (Safety Population)	Safety Population
3	Table 14.1.3.1: Duration of Treatment Exposure (Safety Population)	Safety Population

The left sidebar shows a "Table of Contents" for "demo1" with a list of tables. The bottom of the screen shows a Windows taskbar with various application icons and a system clock indicating 2:44 PM on 8/14/2025.



Video link: <https://www.loom.com/share/70e39fa1d1824b3c91b2395236e47e55?sid=e8584776-b1a8-4446-a64a-df7d240205ea>

Clymb's TFL Automation Solutions & Workflow

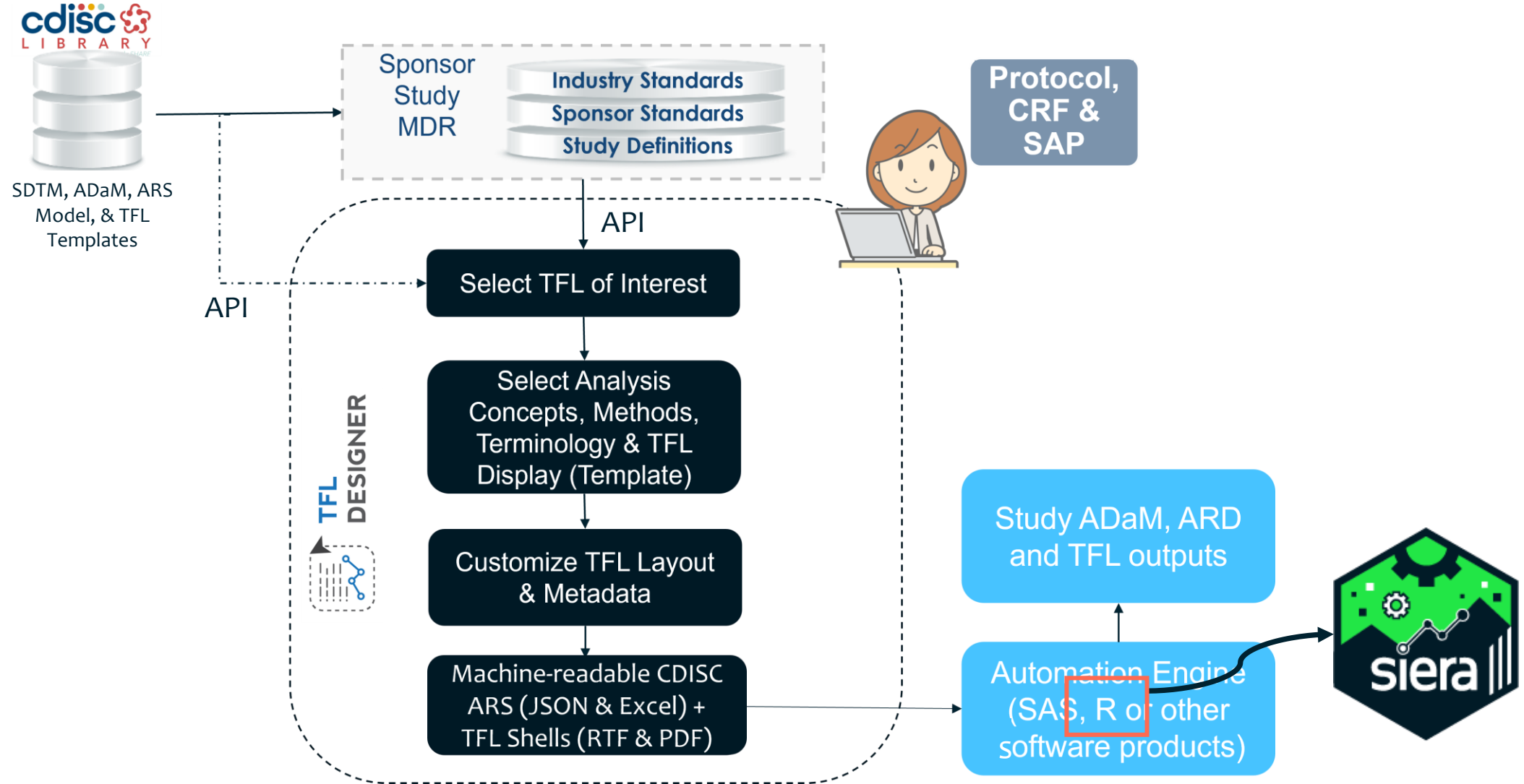


 = Automated Process

TFL Code Generator



Metadata-driven TFL Deliverable Workflow



What is siera?

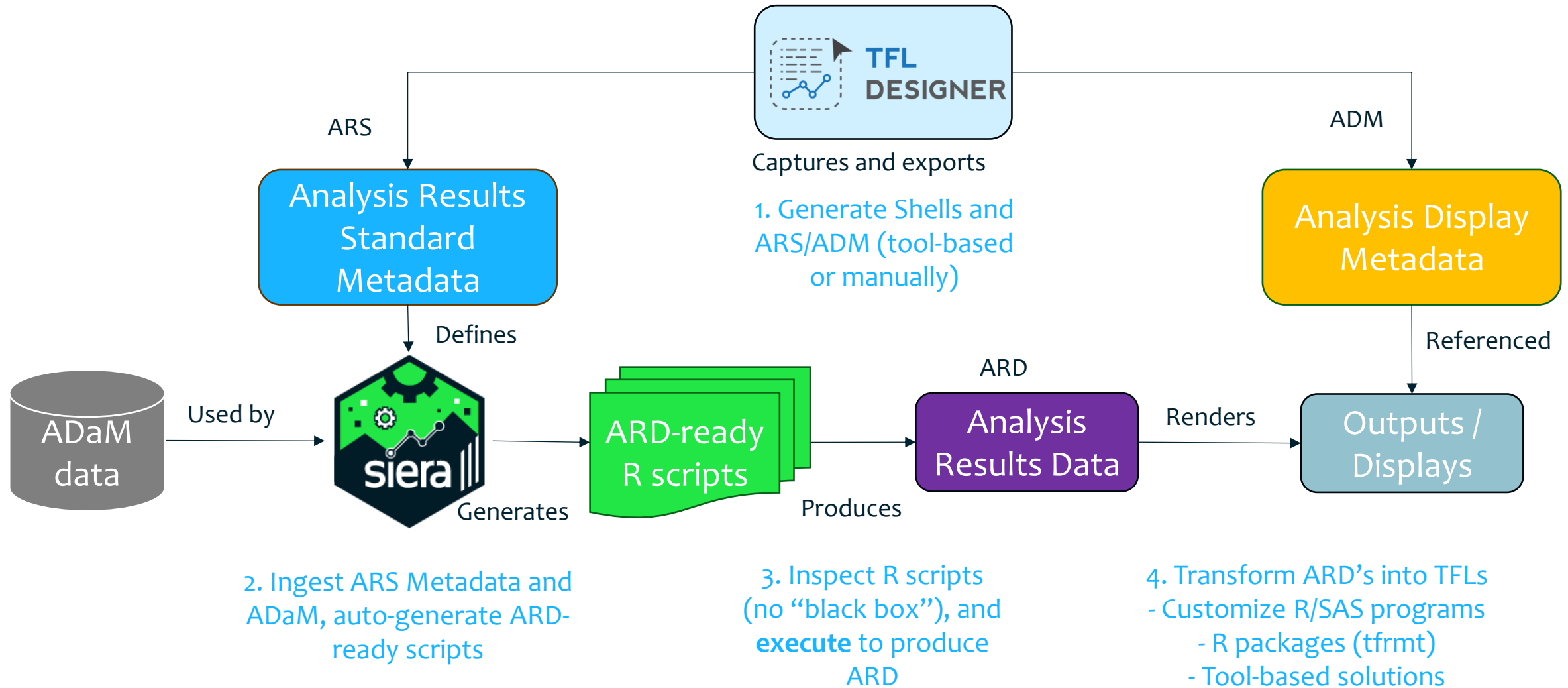
siera* is an open-source **R package** that **ingests ARS metadata** for a reporting event and converts the metadata to **produce executable R scripts**. These R scripts can be run as-is (combined with the relevant ADaM dataset) to **produce each output's ARD**.

<https://github.com/clymbclinical/siera>



* on CRAN and approved by COSA!

siera Workflow [Design to Execution]




```

1 # Import the necessary packages
2 import pandas as pd
3 import numpy as np
4 import matplotlib.pyplot as plt
5 import seaborn as sns
6 import os
7
8 # Set the random seed for reproducibility
9 np.random.seed(42)
10
11 # Create a list of patient IDs
12 patient_ids = ['P001', 'P002', 'P003', 'P004', 'P005', 'P006', 'P007', 'P008', 'P009', 'P010', 'P011', 'P012', 'P013', 'P014', 'P015', 'P016', 'P017', 'P018', 'P019', 'P020']
13
14 # Create a DataFrame for patient data
15 patient_data = pd.DataFrame({
16     'Patient ID': patient_ids,
17     'Age': np.random.randint(20, 80, size=len(patient_ids)),
18     'Sex': np.random.choice(['Male', 'Female'], size=len(patient_ids)),
19     'Weight (kg)': np.random.randint(50, 100, size=len(patient_ids)),
20     'Height (cm)': np.random.randint(150, 200, size=len(patient_ids)),
21     'Blood Pressure (mmHg)': np.random.randint(90, 180, size=len(patient_ids)),
22     'Heart Rate (bpm)': np.random.randint(60, 100, size=len(patient_ids)),
23     'Glucose (mg/dL)': np.random.randint(70, 150, size=len(patient_ids)),
24     'Cholesterol (mg/dL)': np.random.randint(100, 300, size=len(patient_ids)),
25     'Hemoglobin (g/dL)': np.random.randint(12, 18, size=len(patient_ids)),
26     'Creatinine (mg/dL)': np.random.randint(0.5, 2.0, size=len(patient_ids)),
27     'Urea Nitrogen (mg/dL)': np.random.randint(5, 20, size=len(patient_ids)),
28     'Sodium (mEq/L)': np.random.randint(130, 150, size=len(patient_ids)),
29     'Potassium (mEq/L)': np.random.randint(3.0, 5.0, size=len(patient_ids)),
30     'Calcium (mg/dL)': np.random.randint(8.0, 10.0, size=len(patient_ids)),
31     'Phosphorus (mg/dL)': np.random.randint(2.0, 5.0, size=len(patient_ids)),
32     'Magnesium (mg/dL)': np.random.randint(0.5, 1.5, size=len(patient_ids)),
33     'Vitamin D (ng/mL)': np.random.randint(10, 100, size=len(patient_ids)),
34     'Folate (ng/mL)': np.random.randint(5, 20, size=len(patient_ids)),
35     'Vitamin B12 (pg/mL)': np.random.randint(100, 1000, size=len(patient_ids)),
36     'Iron (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
37     'Zinc (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
38     'Copper (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
39     'Manganese (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
40     'Selenium (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
41     'Chromium (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
42     'Molybdenum (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
43     'Cobalt (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
44     'Nickel (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
45     'Vanadium (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
46     'Manganese (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
47     'Selenium (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
48     'Chromium (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
49     'Molybdenum (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
50     'Cobalt (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
51     'Nickel (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
52     'Vanadium (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
53     'Manganese (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
54     'Selenium (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
55     'Chromium (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
56     'Molybdenum (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
57     'Cobalt (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
58     'Nickel (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
59     'Vanadium (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
60     'Manganese (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
61     'Selenium (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
62     'Chromium (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
63     'Molybdenum (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
64     'Cobalt (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
65     'Nickel (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
66     'Vanadium (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
67     'Manganese (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
68     'Selenium (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
69     'Chromium (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
70     'Molybdenum (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
71     'Cobalt (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
72     'Nickel (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
73     'Vanadium (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
74     'Manganese (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
75     'Selenium (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
76     'Chromium (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
77     'Molybdenum (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
78     'Cobalt (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
79     'Nickel (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
80     'Vanadium (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
81     'Manganese (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
82     'Selenium (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
83     'Chromium (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
84     'Molybdenum (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
85     'Cobalt (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
86     'Nickel (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
87     'Vanadium (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
88     'Manganese (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
89     'Selenium (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
90     'Chromium (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
91     'Molybdenum (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
92     'Cobalt (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
93     'Nickel (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
94     'Vanadium (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
95     'Manganese (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
96     'Selenium (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
97     'Chromium (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
98     'Molybdenum (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
99     'Cobalt (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
100     'Nickel (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
101     'Vanadium (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
102     'Manganese (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
103     'Selenium (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
104     'Chromium (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
105     'Molybdenum (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
106     'Cobalt (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
107     'Nickel (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
108     'Vanadium (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
109     'Manganese (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
110     'Selenium (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
111     'Chromium (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
112     'Molybdenum (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
113     'Cobalt (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
114     'Nickel (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
115     'Vanadium (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
116     'Manganese (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
117     'Selenium (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
118     'Chromium (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
119     'Molybdenum (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
120     'Cobalt (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
121     'Nickel (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
122     'Vanadium (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
123     'Manganese (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
124     'Selenium (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
125     'Chromium (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
126     'Molybdenum (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
127     'Cobalt (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
128     'Nickel (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
129     'Vanadium (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
130     'Manganese (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
131     'Selenium (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
132     'Chromium (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
133     'Molybdenum (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
134     'Cobalt (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
135     'Nickel (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
136     'Vanadium (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
137     'Manganese (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
138     'Selenium (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
139     'Chromium (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
140     'Molybdenum (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
141     'Cobalt (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
142     'Nickel (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
143     'Vanadium (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
144     'Manganese (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
145     'Selenium (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
146     'Chromium (mcg/L)': np.random.randint(50, 200, size=len(patient_ids)),
```

Auto-generated R scripts

```
1 # Programme:      Generate code to produce AR for Out14-1-
2 # Output:         Summary of Demographics
3 # Date created:   2025-02-03 09:58:07
4
5 # Load libraries ----
6 library(tidyverse)
7 library(readxl)
8 library(splittstackshape)
9 library(readr)
10
11 # Load ADAM ----
12 ADSL <- read_csv('csv_adam/ADSL.csv')
13
14 # Analysis1_An01_05_SAF_Summ_ByTrt ----
15 # Apply Analysis1 Set ---
16 # Analysis1 set : SAF
17 df_An01_05_SAF_Summ_ByTrt <- dplyr::filter(ADSL,
18                                     SAFFL == 'Y')
```

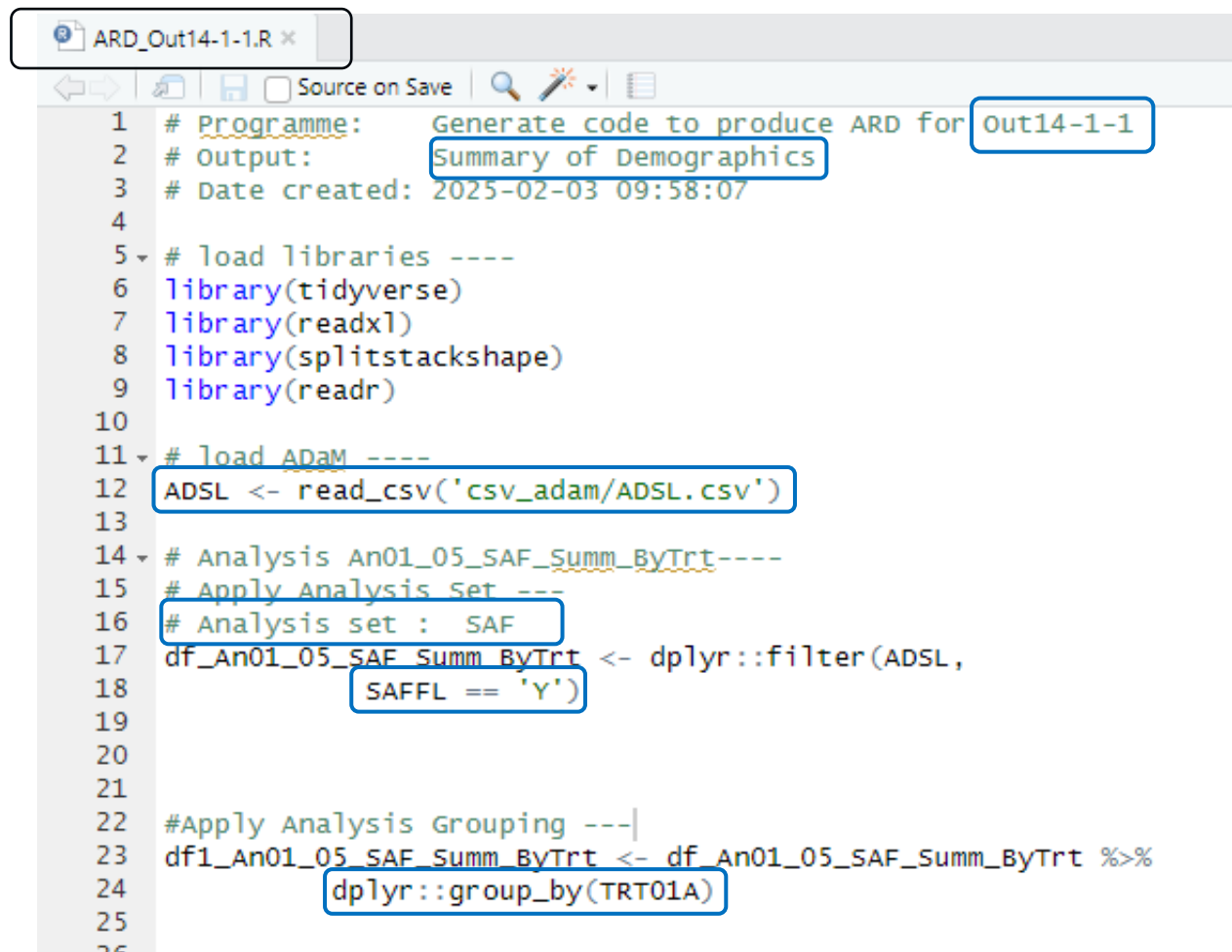
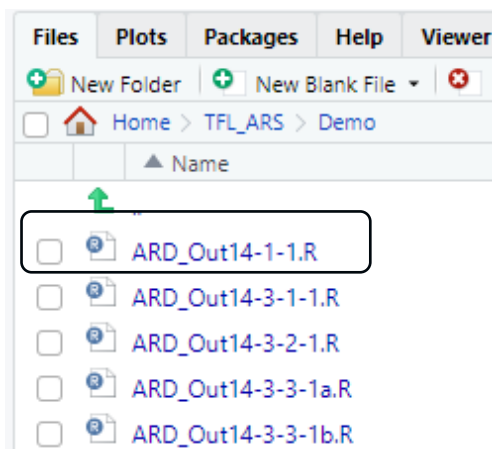
ADaM Dataset

USUBJID	ARM	AGE	AGEGR1	AGEU	RACE	SEX
01-701-1015	Placebo	63	<65	YEARS	WHITE	F
01-701-1023	Placebo	64	<65	YEARS	WHITE	M
01-701-1028	Xanomeline High Dose	71	65+	YEARS	WHITE	M
01-701-1033	Xanomeline Low Dose	74	65+	YEARS	WHITE	M
01-701-1034	Xanomeline High Dose	77	65+	YEARS	WHITE	F
01-701-1047	Placebo	85	65+	YEARS	WHITE	F

Analysis Results Datasets

	operation_id	resultGroup1_groupingId	resultGroup1_groupId	resultGroup2_groupingId	resultGroup2_groupId	rawValu	formattedVal
An03.02_AgeGrp_ByTrt	Mth01_CatVar_ByGrp_1_n	AnlsGrouping_02_Trt	AnlsGrouping_02_Trt_1	AnlsGrouping_03_AgeGp	AnlsGrouping_03_AgeGp_1	14	14
	operation_id	resultGroup1_groupingId	resultGroup1_groupId	resultGroup2_groupingId	resultGroup2_groupId	rawValu	formattedVal
An03.02_AgeGrp_ByTrt	Mth01_CatVar_ByGrp_1_n	AnlsGrouping_02_Trt	AnlsGrouping_02_Trt_1	AnlsGrouping_03_AgeGp	AnlsGrouping_03_AgeGp_1	14	14
	operation_id	resultGroup1_groupingId	resultGroup1_groupId	resultGroup2_groupingId	resultGroup2_groupId	rawValu	formattedVal
An03.02_AgeGrp_ByTrt	Mth01_CatVar_ByGrp_1_n	AnlsGrouping_02_Trt	AnlsGrouping_02_Trt_1	AnlsGrouping_03_AgeGp	AnlsGrouping_03_AgeGp_1	14	14
An03.02_AgeGrp_ByTrt	Mth01_CatVar_ByGrp_1_n	AnlsGrouping_02_Trt	AnlsGrouping_02_Trt_1	AnlsGrouping_03_AgeGp	AnlsGrouping_03_AgeGp_1	14	14
An03.02_AgeGrp_ByTrt	Mth01_CatVar_ByGrp_1_n	AnlsGrouping_02_Trt	AnlsGrouping_02_Trt_1	AnlsGrouping_03_AgeGp	AnlsGrouping_03_AgeGp_1	72	72
An03.02_AgeGrp_ByTrt	Mth01_CatVar_ByGrp_1_n	AnlsGrouping_02_Trt	AnlsGrouping_02_Trt_2	AnlsGrouping_03_AgeGp	AnlsGrouping_03_AgeGp_1	8	8
An03.02_AgeGrp_ByTrt	Mth01_CatVar_ByGrp_1_n	AnlsGrouping_02_Trt	AnlsGrouping_02_Trt_2	AnlsGrouping_03_AgeGp	AnlsGrouping_03_AgeGp_2	76	76
An03.02_AgeGrp_ByTrt	Mth01_CatVar_ByGrp_1_n	AnlsGrouping_02_Trt	AnlsGrouping_02_Trt_3	AnlsGrouping_03_AgeGp	AnlsGrouping_03_AgeGp_1	11	11
An03.02_AgeGrp_ByTrt	Mth01_CatVar_ByGrp_1_n	AnlsGrouping_02_Trt	AnlsGrouping_02_Trt_3	AnlsGrouping_03_AgeGp	AnlsGrouping_03_AgeGp_2	73	73
An03.02_AgeGrp_ByTrt	Mth01_CatVar_ByGrp_2_pct	AnlsGrouping_02_Trt	AnlsGrouping_02_Trt_1	AnlsGrouping_03_AgeGp	AnlsGrouping_03_AgeGp_1	16.27907	(16.3)
An03.02_AgeGrp_ByTrt	Mth01_CatVar_ByGrp_2_pct	AnlsGrouping_02_Trt	AnlsGrouping_02_Trt_1	AnlsGrouping_03_AgeGp	AnlsGrouping_03_AgeGp_2	83.72093	(83.7)
An03.02_AgeGrp_ByTrt	Mth01_CatVar_ByGrp_2_pct	AnlsGrouping_02_Trt	AnlsGrouping_02_Trt_2	AnlsGrouping_03_AgeGp	AnlsGrouping_03_AgeGp_1	9.52381	(9.5)
An03.02_AgeGrp_ByTrt	Mth01_CatVar_ByGrp_2_pct	AnlsGrouping_02_Trt	AnlsGrouping_02_Trt_2	AnlsGrouping_03_AgeGp	AnlsGrouping_03_AgeGp_2	90.47619	(90.5)
An03.02_AgeGrp_ByTrt	Mth01_CatVar_ByGrp_2_pct	AnlsGrouping_02_Trt	AnlsGrouping_02_Trt_3	AnlsGrouping_03_AgeGp	AnlsGrouping_03_AgeGp_1	13.09524	(13.1)
An03.02_AgeGrp_ByTrt	Mth01_CatVar_ByGrp_2_pct	AnlsGrouping_02_Trt	AnlsGrouping_02_Trt_3	AnlsGrouping_03_AgeGp	AnlsGrouping_03_AgeGp_2	86.90476	(86.9)

siera generated ARD-ready R scripts



siera Demo Video

The screenshot displays the RStudio interface with a script editor on the left, an environment pane on the right, and a console at the bottom. The script, named 'demo14AUG2025.R', contains R code for reading ADaM data and running the 'siera' function. The environment pane shows the 'Global Environment' with no objects. The console shows the command 'run siera' being executed. Below the console, three file explorer windows are visible, showing the directory structure of the project.

```
1 # CDISC example
13 # demo
14
15 ARS_pathj = "C:/Users/mbosm/OneDrive - Clymb Clinical/Documents/demo/14AUG2025/demo1.json"
16 ARS_pathx = "C:/Users/mbosm/OneDrive - Clymb Clinical/Documents/demo/14AUG2025/demo1.xlsx"
17
18 ADaM_csv_path = 'C:/Users/mbosm/OneDrive - Clymb Clinical/Documents/TFL_ARS/csv_adam'
19 ard_programs = "C:/Users/mbosm/OneDrive - Clymb Clinical/Documents/demo/14AUG2025/ard_prog
20
21 # run siera -----
22
23 readARS(ARS_path = ARS_pathx,
24         adam_path = ADaM_csv_path,
25         output_path = ard_programs)
26
27 readARS(ARS_path = ARS_pathj,
28         adam_path = ADaM_csv_path,
29         output_path = ard_programs)
30
```

Environment: Global Environment (240 MiB)

Environment is empty

Files: Home

Name	Size	Modified
Default.rdp	2.4 KB	Aug 14, 2025
desktop.ini	418 B	Aug 13, 2025
.Renvron	55 B	Aug 4, 2025, 10
siera_0.5.0.tar.gz	243.4 KB	Jul 29, 2025, 5:0
.Rhstory	2.6 KB	Jul 21, 2025, 10
siera_0.4.0.tar.gz	1.1 MB	Jul 17, 2025, 12
ISS_testing.xlsx	10.4 KB	Apr 28, 2025, 2



Video link: <https://www.loom.com/share/4aed3442c335446f83d4ba4caa59faf3?sid=b15d1fa5-bb9f-4857-a754-647de14b6bdb>

Impact, Learnings and Next Steps

Impact

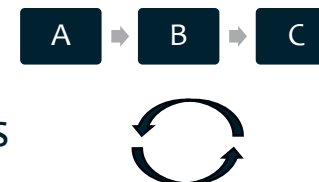
- ↑ Metadata setup
- ↓ Manual programming
- ↑ Time spent on complex tasks
- ↓ Time spent on repetitive tasks
- ↑ Quality and efficiency
- ↓ Overall Cost

Learnings

- Need of a design tool to generate both ARM and ADM metadata
- Integrating ARD into the process requires change
- Avoid “black box” solution
- Provide flexibility to the user

Next Steps

- Integration into workflows
- More feedback, more updates



AI Code Generator

TFL Shells → CDISC ARS Metadata (JSON) from TFL Designer

TFL DESIGNER Neurology / Alzheimer's / CDISC - eTFL Portal

Table Of Contents Baseline Demographic and ...

CDISC - eTFL Portal

Generated using TFL Designer (Community, v1.0)

FDA-DM-T02

Baseline Demographic and Clinical Characteristics

Safety Population

Characteristics	Xanomeline Low Dose (N=XX) n (%)	Xanomeline High Dose (N=XX) n (%)	Placebo (N=XX) n (%)
Sex, n (%)			
Male	XX (XX.X)	XX (XX.X)	XX (XX.X)
Female	XX (XX.X)	XX (XX.X)	XX (XX.X)
Intersex	XX (XX.X)	XX (XX.X)	XX (XX.X)
Unknown	XX (XX.X)	XX (XX.X)	XX (XX.X)
Age, Years			
n	XX	XX	XX
Mean (SD)	XX.X (XX.XX)	XX.X (XX.XX)	XX.X (XX.XX)
Median	XX.X	XX.X	XX.X
Min, Max	XX.X, XX.X	XX.X, XX.X	XX.X, XX.X
Age groups (years), n (%)			

portal-export_Tables.json {} eTFL-portal-export.json

```
{
  "about": {
    "generatedUsing": "TFL Designer Enterprise",
    "version": "1.4.0",
    "note": "The metadata provided herein has been generated using the TFL Designer Enterprise version 1.4.0. The representation of metadata fields av"
  },
  "studyInfo": {
    "studyId": "CDISC - eTFL Portal",
    "studyTitle": "Safety and Efficacy of the Xanomeline Transdermal Therapeutic System (TTS) in Patients with Mild to Moderate Alzheimer's Disease.",
    "phase": "Phase II",
    "compoundUnderStudy": "Xanomeline Transdermal",
    "description": "",
    "diseaseArea": "Neurology",
    "therapeuticArea": "Alzheimer's"
  },
  "name": "Clinical Study Report",
  "id": "CSR",
  "mainListOfContents": {
    "name": "List of Planned Analyses",
    "label": "LOPA",
    "contentsList": {
      "listItems": [
        {
          "name": "Baseline Demographic and Clinical Characteristics",
          "level": 1,
          "order": 1,
          "outputId": "Out_01",
          "sublist": {
            "listItems": [
              {
                "name": "Summary of Subjects by Treatment",
                "level": 2,
                "order": 1,
                "analysisId": "An_01"
              },
              {
                "name": "Sex, n (%)",
                "level": 2,
                "order": 2,
                "sublist": {
                  "listItems": [
                    {
                      "name": "Summary of Subjects by Treatment and Sex, n (%)",
                      "level": 3,
                      "order": 1,
                      "analysisId": "An_02"
                    }
                  ]
                }
              }
            ]
          }
        }
      ]
    }
  }
}
```

AI Code Generator for TFL Automation



TFL
DESIGNER

+

✦ ✦ **AI Code Generator**

Transforming TFL shells and metadata
into ready-to-use SAS and R code

Introducing AI Code Generator for TFL Automation



TFL
DESIGNER

+

AI Code Generator

*Transforming your TFL shells and metadata into
ready-to-run SAS and R code in seconds*

AI Code Generator



Video link: <https://www.loom.com/share/90c1fa1929b54c4b83e1fdf1e578dc04?sid=92589371-8b75-4242-b0e2-bc7c74e3aab5>

Q&A





Anna Yaggi

Product Manager

ayaggi@clymbclinical.com

Shivani Gupta

Director of Statistical Programming

sgupta@clymbclinical.com

Malan Bosman

Lead Automation Engineer

mbosman@clymbclinical.com



EMAIL

info@clymbclinical.com



WEBSITE

www.clymbclinical.com



PHONE

781-692-3613