

Do EudraCT the Easy Way: Semi-Automation of Inputting Study & Treatment Event Information in EudraCT

Kevin Miller & Larry Sleeper

Author Bios



Kevin Miller is a Manager of Statistical Programming at Gilead Sciences. Kevin's team leads data engineering projects to automate and optimize processes throughout the organization



Larry Sleeper is a Solution Architect for Catalyst Clinical Research offering programming support and vision to the Data Engineering team at Gilead. Larry is lead programmer of the EudraCT automation tool

Agenda

- What is EudraCT?
- Python Solution
 - Modules
 - Process
- Input
 - Excel
 - XML
- Your Turn
- Credits / Disclaimers

What is EudraCT?

- European **U**nion **D**rug **R**egulating **A**uthorities **C**linical **T**rials
- European database for all interventional clinical trials on medicinal products authorized in the European Union.
- Tool for searching for clinical trials
- Promotes Transparency
- Sponsors have requirement of uploading trial information in a timely manner
- European equivalent of [ClinicalTrials.gov](https://clinicaltrials.gov)

Successor to Eudract - CTIS

- The Clinical Trial Information System (CTIS) is the successor to the current European trial registry EudraCT. Planned launch is late 2021.
- From late 2021 onwards, all new drug trials (CTIMPs) starting up in Europe will have to be registered on CTIS rather than on EudraCT.
- However, CTIS will not immediately replace EudraCT. Instead, **both systems will run in parallel**. Sponsors will have to continue managing their clinical trials already registered on EudraCT, including updating their status and uploading results onto EudraCT.
- Source : <https://www.transparimed.org/single-post/clinical-trial-information-system-ctis-training>

Adverse Events Reporting

- Safety data of adverse events, including counts, and individual events categorized on treatment groups.
- Distinct Sections:
 - Treatment Groups
 - Serious Events
 - Non-Serious Events

EudraCT Web User Interface

Adverse event reporting group definition

Reporting group title:*

Group A 

Reporting group description:

Group A Reporting 

For this reporting group, provide the following totals:

Subjects exposed: * 

Subjects affected by serious adverse events: * 

Subjects affected by non-serious adverse events: * 

Total number of deaths (all causes): * 

Total number of deaths resulting from adverse events: 



Automation

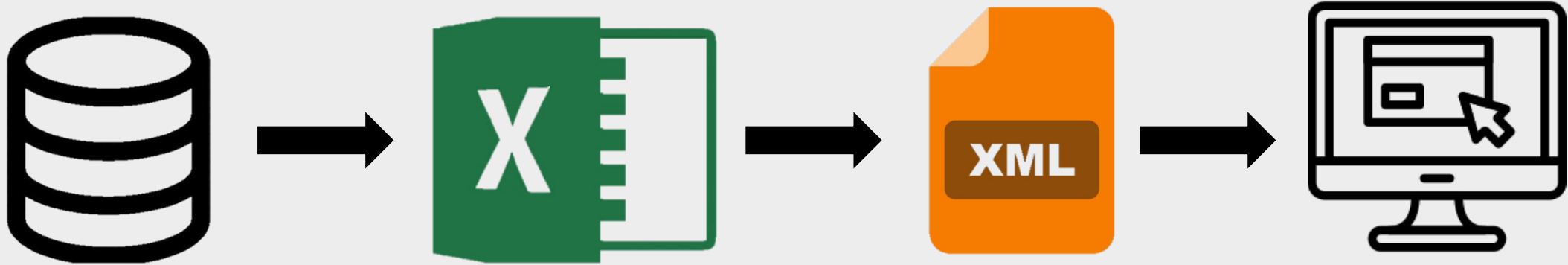
- Build on Clinical Data Transparency team efforts to maintain CTGov and EudraCT for Gilead studies
- Use of existing Biometrics SAS generated outputs to create Adverse Event XML file for upload
- User can upload a single XML file per study, instead of manual data entry in EudraCT web UI.
- Inspired by R package “eudract” - <https://cran.r-project.org/web/packages/eudract/index.html>

Python Solution

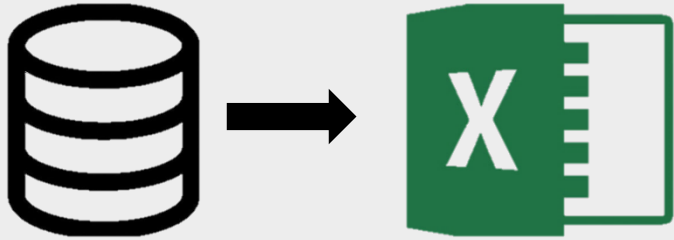
- Leverage popular Python modules
 - pandas
 - Ubiquitous data handling python module, easy importing of excel files, and manipulating data in an excel/SAS like method with dataframes.
 - The pandas module is used to import Excel data into pandas **dataframes**
 - The dataframes are then iterated over to create XML (like a data step in SAS)
 - lxml
 - lxml is the most feature-rich and easy-to-use module for processing XML and HTML in the Python language.
 - lxml has methods to build XML files for upload that meet the requirements of EudraCT Adverse Event XML



Process Flow



Process Flow – Excel Spreadsheet



- Reports of Safety data from clinical trials are created by Biometrics in Excel
- Spreadsheets are created for both serious and non-serious safety events.

	A	B	C	D	E	F
1	System_Organ_Class ▾	Preferred_Term ▾	Treatment_Groups ▾	Subjects_Affected_Number ▾	Subjects_Exposed_Number ▾	Occurrences_All_Number ▾
2			Placebo	20	137	
3			Study Drug	55	142	
4	Gastrointestinal disorders	Colitis ulcerative	Placebo	5	137	5
5	Gastrointestinal disorders	Abdominal pain	Placebo	3	137	3
6	Gastrointestinal disorders	Abdominal pain	Study Drug	9	142	9
7	Gastrointestinal disorders	Nausea	Placebo	1	137	1
8	Gastrointestinal disorders	Nausea	Study Drug	6	142	6
9	Infections and infestations	Nasopharyngitis	Placebo	2	137	2
10	Infections and infestations	Nasopharyngitis	Study Drug	11	142	11
11	Infections and infestations	URI	Placebo	0	137	0
12	Infections and infestations	URI	Study Drug	5	142	6
13	Nervous system disorders	Headache	Placebo	6	137	8
14	Nervous system disorders	Headache	Study Drug	9	142	13

Process Flow – Build XML



- Python lxml module handles XML header, Elements, parent child relationships, and XML namespaces.
- Much easier than building with strings in SAS

```
0 10 20 30 40 50 60 70 80 90 100
1 <?xml version='1.0' encoding='UTF-8'?>
2 <ae:adverseEvents xmlns:ae="http://eudract.ema.europa.eu/schema/clinical_trial_result/adverse_events"
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
4   <nonSeriousEventFrequencyThreshold>5</nonSeriousEventFrequencyThreshold>
5 </ae:adverseEvents>
```

```
0 10 20 30 40 50 60 70 80 90 100
1 # set up XML namespaces necessary EudraCT XML file
2 ns_eudra_ae="http://eudract.ema.europa.eu/schema/clinical_trial_result/adverse_events"
3 ns_xsi="http://www.w3.org/2001/XMLSchema-instance"
4
5 # create dictionary of XML namespace mappings
6 NSMAP = { "ae":ns_eudra_ae, "xsi":ns_xsi}
7
8 # create root adverseEvents element and nonSeriousEventFrequencyThreshold as child
9 ae=etree.Element("{%s}adverseEvents" % (ns_eudra_ae), nsmap = NSMAP)
10 threshold=etree.SubElement(ae,"nonSeriousEventFrequencyThreshold")
11
12 # non-serious event threshold is always 5
13 threshold.text="5"
```

Create XML – Treatment Group Counts



Below is the XML created to represent treatment group counts

```
0 10 20 30 40 50 60 70 80 90 100
1 <reportingGroups>
2   <reportingGroup id="ReportingGroup-0">
3     <title>Placebo</title>
4     <description>Patients who received Placebo</description>
5     <subjectsAffectedByNonSeriousAdverseEvents>40</subjectsAffectedByNonSeriousAdverseEvents>
6     <subjectsAffectedBySeriousAdverseEvents>3</subjectsAffectedBySeriousAdverseEvents>
7     <subjectsExposed>245</subjectsExposed>
8     <deathsAllCauses>0</deathsAllCauses>
9     <deathsResultingFromAdverseEvents>0</deathsResultingFromAdverseEvents>
10  </reportingGroup>
11 </reportingGroups>
```

```
0 10 20 30 40 50 60 70 80 90 100
1 # create reportingGroups node
2 repgroups_node=etree.SubElement(node,"reportingGroups")
3
4 # iterate reporting group dataframe and create necessary subelements
5 for index, row in df_counts.iterrows():
6     repgroupnode=etree.SubElement(repgroups_node,"reportingGroup")
7     repgroupnode.set("id","ReportingGroup-"+str(index))
8     title=etree.SubElement(repgroupnode,"title")
9     rpname=row["Treatment_Groups"]
10    description=etree.SubElement(repgroupnode,"description")
11    description.text="Patients who received "+rpname
12    subs_affected_nsae=etree.SubElement(repgroupnode,"subjectsAffectedByNonSeriousAdverseEvents")
13    subs_affected_nsae.text=str(int(row["NSAE_Subjects_Affected"]))
14
```

Create XML – SAE frequency



Below is the XML created to represent serious adverse event counts

SAE and NSAE XML are very similar. Python code to build this XML is in next slide

```
0 10 20 30 40 50 60 70 80 90 100
1 <seriousAdverseEvents>
2   <seriousAdverseEvent>
3     <term>Anaemia</term>
4     <organSystem>
5       <eutctId>100000004851</eutctId>
6       <version>22</version>
7     </organSystem>
8     <values>
9       <value reportingGroupId="ReportingGroup-0">
10         <occurrences>0</occurrences>
11         <subjectsAffected>0</subjectsAffected>
12         <subjectsExposed>245</subjectsExposed>
13         <occurrencesCausallyRelatedToTreatment>0</occurrencesCausallyRelatedToTreatment>
14         <fatalities>
15           <deaths>0</deaths>
16           <deathsCausallyRelatedToTreatment>0</deathsCausallyRelatedToTreatment>
17         </fatalities>
18       </value>
19     </seriousAdverseEvent>
20 </seriousAdverseEvents>
```

Create XML – NSAE frequency



```
1 <nonSeriousAdverseEvents>
2   <nonSeriousAdverseEvent>
3     <term>Anaemia</term>
4     <organSystem>
5       <eutctId>100000004851</eutctId>
6       <version>22</version>
7     </organSystem>
8     <values>
9       <value reportingGroupId="ReportingGroup-0">
10        <occurrences>7</occurrences>
11      </value>
12    </values>
13   <nonSeriousAdverseEvent>
14 </nonSeriousAdverseEvents>
```

```
1 # non-serious adverse events
2 nsaes=etree.SubElement(node,"nonSeriousAdverseEvents")
3 for index, row in df_terms.iterrows():
4   organ_system = row["System_Organ_Class"]
5   nsae=etree.SubElement(nsaes,"nonSeriousAdverseEvent")
6   term=etree.SubElement(nsae,"term")
7   term.text= row["Preferred_Term"]
8   soc=etree.SubElement(nsae,"organSystem")
9   eutctid=etree.SubElement(soc,"eutctId")
10  eutctid.text=str(spor_id[organ_system])
11  version=etree.SubElement(soc,"version")
12  version.text=str(spor_term_version[organ_system])
13  values=etree.SubElement(nsae,"values")
14  value=etree.SubElement(values,"value")
15  rpname=row["Treatment_Groups"]
16  value.set("reportingGroupId","ReportingGroup-"+reporting_groups[row["Treatment_Groups"]])
17  occurrences=etree.SubElement(value,"occurrences")
18  occurrences.text=str(int(row["Occurrences_All_Number"]))
19
```

SPOR Codes

EudraCT has their own codes they use to classify Body Systems.

The SPOR Referential Management System is where to find those codes.

<https://spor.ema.europa.eu/rmswi/#/>

The body system categories need to be mapped to their appropriate SPOR code.

We store them in a Python dictionary and look them up based upon SOC (System Organ Class).

	A	B	C
1	SPOR_identifier	Body_System	Version
2	100000004848	Investigations	22
3	100000004849	Cardiac disorders	22
4	100000004850	Congenital, familial and genetic disorders	22
5	100000004851	Blood and lymphatic system disorders	22
6	100000004852	Nervous system disorders	22
7	100000004853	Eye disorders	22
8	100000004854	Ear and labyrinth disorders	22
9	100000004855	Respiratory, thoracic and mediastinal disorders	22
10	100000004856	Gastrointestinal disorders	22
11	100000004857	Renal and urinary disorders	22
12	100000004858	Skin and subcutaneous tissue disorders	22

Upload XML to EudraCT



- Upload XML via Eudra CT.
- Ensure option to only update the Adverse Events portion
- If XML is not formed correctly you will be given Error message.
- EudraCT training site allows you to test uploads without affecting public website

Upload results from an XML file

Add XML file containing the results

Type of file: Adverse Events ▼

Full Results Set

Adverse Events

+ Add...

Upload

Return to index

Implementation Review

- Using the automation processes is significantly faster.
 - ~1 hour vs 20+ hours previously.
- Eliminates the possibility of data entry mistakes.
- Forces our organization to standardize the process from end to end.

Your Turn

If you are interested in using this tool visit our GitHub page: https://github.com/kevinjacksonm/eudract_ae

The screenshot displays the GitHub repository page for `kevinjacksonm/eudract_ae`. The repository is on the `main` branch, has 2 branches, and 0 tags. It shows a commit history with 21 commits. The file list includes `LICENSE`, `README.md`, `SPOR_lookup.xlsx`, `eudract_ae.py`, and `example_xlsx_template.xlsx`. The README content is visible, featuring the EudraCT logo and a description of the Python source code used to create XML adverse events files for upload to the EudraCT website. It mentions that it uses `.xlsx` as source input and provides a link to a sample template file. A table of sample data is also shown.

	A	B	C	D	E	F
1	System_Organ_Class	Preferred_Term	Treatment_Groups	Subjects_Affected_Number	Subjects_Exposed_Number	Occurrences_All_Number
2			Placebo	20	137	
3			Study Drug	55	142	
4	Gastrointestinal disorders	Colitis ulcerative	Placebo	5	137	5
5	Gastrointestinal disorders	Abdominal pain	Placebo	3	137	3
6	Gastrointestinal disorders	Abdominal pain	Study Drug	9	142	9
7	Gastrointestinal disorders	Nausea	Placebo	1	137	1
8	Gastrointestinal disorders	Nausea	Study Drug	6	142	6
9	Infections and infestations	Nasopharyngitis	Placebo	2	137	2
10	Infections and infestations	Nasopharyngitis	Study Drug	11	142	11
11	Infections and infestations	URI	Placebo	0	137	0
12	Infections and infestations	URI	Study Drug	5	142	6
13	Nervous system disorders	Headache	Placebo	6	137	8
14	Nervous system disorders	Headache	Study Drug	9	142	13

- Source Code
- Sample Template
- SPOR Codes
- Read Me

Final Thoughts

- Reliant on having a Python environment.
- Requires collaboration between multiple groups.
- Requires Standard XLSX Output.
- CTIS will be replacing this portal in the near future.

Credits

- To our colleagues at Gilead who worked to improve this process.
- Our managers for leadership and guidance.
- EudraCT for being helpful and responding to our questions.