Paper SM03

Standardized and Automated Template for Dose Escalation Decision

Jasmina Uzunovic, Hoffman-La Roche Canada, Mississauga, Canada
Aditi Qamra, Hoffman-La Roche Canada, Mississauga, Canada
Nina Qi, Genentech, South San Francisco, USA

ABSTRACT

Dose escalation (DE) decision making is a routine process in Phase I studies, which involves thorough review of safety data collected at current dose levels, including monitoring of adverse events, vital signs and laboratory data. First DE meetings can occur shortly after study start-up, where ADaM datasets are often not yet available, leading to various adhoc solutions by study teams to present the data. This leads to lack of standardization in structure, content and format of the slides used across and within therapeutic areas. We will present an R-based solution to automatically populate slide decks with standardized text, templates, and outputs to improve efficiency for this decision making process.

INTRODUCTION

Dose escalation (DE) meetings are critical decision making points in early phase trials. These meetings typically are led by clinical scientists showcasing patient safety and optionally efficacy data from each dose cohort. The analysis and outputs used by clinical scientists can be either generated in collaboration with biometric teams (biostatisticians and programmers) or derived ad hoc from their medical data review. Depending on the stage of the study and criticality of the decision e.g. safety evaluation of a given cohort vs dose optimization decision, the content and format of the dose escalation meeting decks can be quite varied. Additionally, agnostic to the method by which outputs are generated, clinical and biometric teams spend significant time populating numbers from static outputs to powerpoint presentations, risking typographical errors and decreasing overall efficiency.

The aforementioned challenges necessitates the development of a standardized and automated dose escalation slide deck template that can be programmatically populated and is consistent across cohorts and trials. In order to achieve a FAIR DE meeting deck, we kept three main considerations in mind - data source, format of outputs and ease of implementation.

Data sources:

Both SDTM and ADaM datasets were evaluated as data sources for generating the content of the slide decks. However early stage studies do not always have ADaM datasets available at the time of the first few DE meetings. SDTMs datasets are programmed first and key datasets including demographics, disposition and adverse events are almost always prioritized for programming. We thus choose to move forward with SDTMs as the data source for our programming.

Format:

We included both patient level data i.e listings as well as dose and patient level aggregate summaries in the list of outputs to be generated for DE meeting decks. Patient level summaries included Demographics, Baseline characteristics, Exposure summaries, Adverse Events (AE), Vital signs, medical history and Prior cancer therapy & Concomitant medications. Dose level summaries included Disposition Summaries, AE overview, Summary of all AEs, related AEs & G3-5 AEs and summary of fatal events.

Implementation:

While deciding on the programming methodology to implement the decided formats, we kept in mind the following considerations. The code and auto-generated slide deck should have a quick turnaround time (few hours) from SDTM availability. The code should be generalizable across studies and indications, scaling well with increase in the number of cohorts. Finally, our solution should be flexible enough to incorporate new outputs as study demands increase.

METHODOLOGY

In order to implement an automated slide deck solution, we decided to leverage `autoslideR`, an internal R package which automates the creation of slide decks from clinical outputs. There is also an external version of the `autoslideR` package, called `autoslider.core`, which is available on CRAN[1].

The package `autoslideR` takes as input ADaM datasets and human readable metadata files, uses NEST[2] package functions or custom user-built functions to produce TLGs as R objects, and then uses the R package `officer`[3] to produce an output powerpoint document (pptx) containing the TLGs, along with titles and footnotes.

For our purposes, as we want to use SDTM datasets instead of ADaM, we extended the `autoslideR` framework, creating our own custom functions which produce TLGs based off of SDTM data. We also introduce another metadata file, metadata.yaml, to hold extra information which is used to populate the final slide deck adding to reproducibility and traceability of the data used to populate the slide deck (see Figure 1).

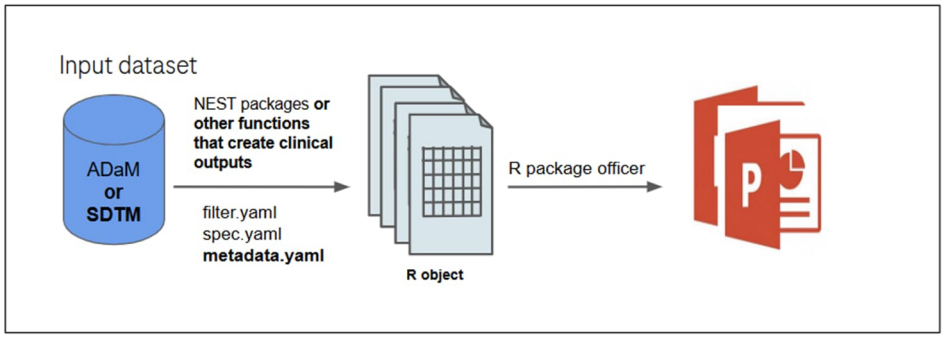

Figure 1 - `autoslideR` workflow, text in bold are add-ons we developed for the dose escalation template

METADATA YAML FILES

As mentioned above, `autoslideR` uses human readable yaml files to drive automation and minimize R programming efforts. There are three main files that are used: filters.yaml, spec.yaml and metadata.yaml. We include examples of all of them below. In figure 2 you can see examples from filters.yaml. On top, there is a filter definition for patient 01 on the DM domain. Separate filters are created for every SDTM domain, and for every patient. There are also filters which are more akin to those used on ADaM datasets, which you can see an example of on the bottom of figure 2. The 'REL' filter is defined for AEs related to study treatment, however here the filter is applied to the AE SDTM domain.

```
PP01:
     title: Patient 01
     condition: grepl("01", USUBJID)
     target: dm
```

```
REL:
     title: Related AEs
     condition: AEREL == "Y"
     target: ae
```

Figure 2 - example of filters defined in the filter.yaml metadata file

The spec.yaml metadata file defines each output that will be included in the slide deck. The program is the function to be used to create the output, which can be one of the functions included with the `autoslideR` package or a custom function built by the user. The suffix is one of the filters defined in filters.yaml. The user can also use spec.yaml to define arguments that will be passed into the program function. In figure 3 we include an example of four outputs defined in the spec.yaml. In the top box, we define a cohort overview listing, which uses argument 'dltwin' for the length of the dose limiting toxicity window, which here we define as 21 days. The 'DM' filter specifies that the DM domain should be used for this output. In the middle box, we define two patient level listings, to create a listing with the patient's baseline disease characteristics and a listing with medical history. In the bottom box, we define a safety summary table filtered for treatment-related AEs, using the DM, EX and AE domains.

```yaml
- program: dm_cohort_list_slide
  suffix: DM
  args:
    dltwin: 21
```

```yaml
- program: df_dm_disease_slide
  suffix: PP01
- program: df_mh_slide_gred
  suffix: MHPP01
```

```yaml
- program: t_ae_sum_g35_sdtm_slide
  suffix: REL_DM_EX_AE
```

Figure 3 - example of specs defined in the spec.yaml metadata file

We introduced the metadata.yaml file to collect information such as molecule, study ID and cohort ID which can be incorporated throughout the slide deck, as well as information important for traceability and reproducibility, such as data extraction date and code gitlab repo link, which will be added to a slide at the end of the deck. You can see an example in figure 4.

```yaml
data_path: data/sdtmv/
ro: RO1234567
study_id: PR12345
cut_date: 17-JUN-2024
extr_date: 17-JUN-2024
mtg_date: 21-JUN-2024
cohort_value: Cohort 1
spotfire_link: <link>
gitlab_repo: <link>
```

Figure 4 - example of metadata.yaml file

RUNNING THE CODE

Once metadata yaml files are complete, and any custom functions have been created, then the main code can be run. The first step is to read in the metadata yaml files and the SDTM domains:

```r
# Read in yaml files
filters::load_filters("metadatain/tlg/dose_filters.yml")
spec_file <- file.path("metadatain/tlg/dose_spec.yml")
metadata <- yaml::read_yaml(file.path("metadatain/tlg/dose_metadata.yml"))
# Get path to data from metadata yaml
data_path <- metadata$data_path
# Read in SDTM
dm <- arrow::read_parquet(paste0(data_path, "dm.parquet"))
ae <- arrow::read_parquet(paste0(data_path, "ae.parquet"))
# … and all other SDTM domains
```

Next, all the input SDTM domains need to be in a named list. Any custom functions that the user creates, need to be sourced. Custom functions will have the named list of SDTM domains as input, along with any arguments that are defined in the spec.yaml file, and will return a data frame or rtables object (from the `rtables` package[4]), which will be the output TLG.

```r
# define the list of input data required for slides
de_data <- list(
  "dm" = dm,
  "ae" = ae
  # … and all other SDTM domains
)
# Source template functions
```

```
lapply(list.files("./R/"), function(x) source(paste0("./R/", x)))
```

Once everything is loaded, there are a few main function calls to create the slide deck. First, read_spec() will load the information from the spec.yaml file into a list (this is the metadata file which includes the specifications for each output). Then generate_outputs() will use the spec list as well as the input SDTM list to create all the outputs as data frame/rtables objects. It will also use the filters in each output spec to filter the input SDTM domain before passing the domain to the function which creates the output object. Next, decorate_outputs() will add titles and footnotes to all the output objects. Finally, generate_slides() will add each output object to a new slide, all in the same slide deck.

```
# Run the outputs
# Starting with the spec yaml file
outputs <- spec_file %>%
  read_spec() %>%
  # create the TLGs with the list of SDTM domains as input
  generate_outputs(datasets = de_data) %>%
  # decorate the TLGs with titles, footnotes
  decorate_outputs() %>%
  # add the TLGs to slides in pptx
  generate_slides(
    outfile = “path/to/outfile”,
    template = file.path(system.file(package = “autoslider.core”),
“/theme/basic.pptx”),
    table_format = autoslider_dose_format
  )
```

Up to this point, we have created a slide deck with all the output objects, saved to ‘outfile’. However, we want to create a more complete slide deck with title slide, agenda, section dividers, slides with headers that clinical science team will fill in the main body text, and a source data slide. We developed a custom function, append_text_slides() which takes as input the data that was defined in the metadata.yaml file, and adds all of these slides to the slide deck created above.

```
# append texts to the slides
# custom function we built to create text slides such as title, section headers,
slides which
# clinical science fills in, final ‘Data Sources’ slide
append_text_slides(
  outfile,
  study_id = metadata$study_id,
  cut_date = metadata$cut_date,
  # … and all other values in metadata yaml
)
```

SLIDE DECK EXAMPLES

We include one example of a final slide from the slide deck in figure 5, showing a listing of AEs in patient 01, along with colour formatting to emphasize AE grade as well as treatment related AEs. At the bottom, in gray you can see a footnote indicating this is a review of Cohort 1.

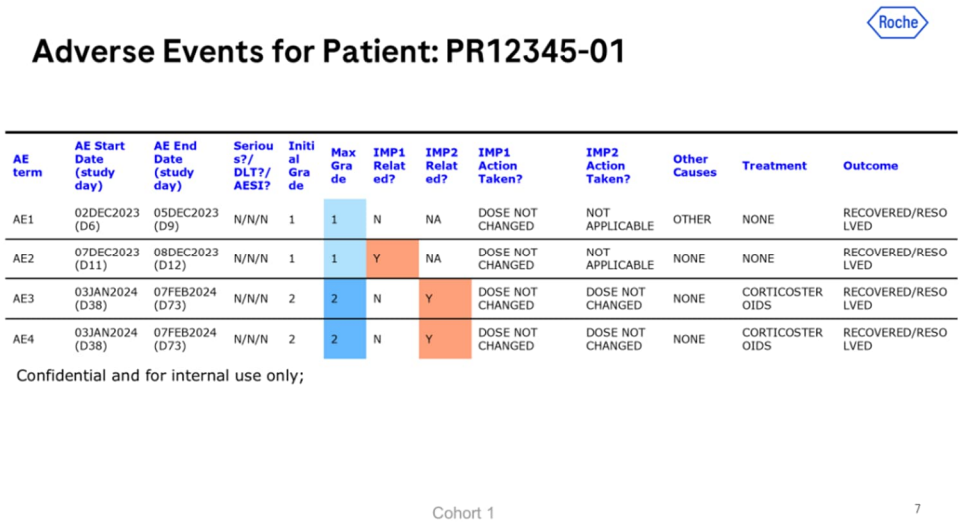

Figure 6 - Example slide from final slide deck showing patient level AE listing

PILOT RESULTS & DISCUSSION

In our pilot experience, using `autoslideR` for our clinical dose escalation (DE) process has been a significant time-saver, reducing the slide preparation time from three days to around five hours for the first DE meeting. For subsequent DE meetings where no major changes are expected, the preparation time can further reduce to just 1-2 hours. Beyond efficiency, the tool has enhanced collaboration and strengthened our influence with stakeholders by bringing greater alignment to the DE process through a more structured and automated approach. However, there are areas where the team can further improve the process. The timing of slide implementation is dependent on SDTM data availability, requiring advanced alignment on resources and timeline within the team. Additionally, formatting enhancements are needed to improve the readability and consistency of the generated slides. There is also a need for clearer guidelines on customization levels for each study's DE meeting templates to balance flexibility with standardization. Lastly, it is crucial for the study teams to align the delivery timeline of the final slide deck in advance, to allow sufficient time for any necessary manual updates before the meeting.

FUTURE STEPS

For the project-level next steps, we plan to conduct one more pilot with either a different DE method or a different team to further refine the approach. The working group (WG) will finalize recommendations on the DE meeting process, template, and timeline, paving the way for a formal roll-out. As a part of this, the Clinical/Safety team will update the process plan, while the Data Science team will update the Wiki page to support future DE study adoption.

On the technical level, we plan to integrate pilot template functions into the both `autoslideR` (internal package) and `autoslider.core` (open-source validated package), as well as expanding the scope with additional functions based on the proposed DE template. To help with the roll-out and wide adoption, we will recommend best practices for the programming team to implement this on our internal platform, as well as improve documentation on how study-specific customizations can be made. Last but not least, we will collaborate with the `autoslideR` team to explore ways to further reduce setup time, such as leveraging yaml files.

CONCLUSION

In summary, the implementation of this `autoslideR`-based automated solution for generating dose escalation (DE) meeting slides has significantly enhanced the efficiency and consistency of data presentation in our pilot early phase study. By leveraging the `autoslideR` package and extending its capabilities with functionalities developed based on the proposed DE templates, we have reduced the slide preparation time from days to hours, minimized typographical errors, and ensured a higher degree of reproducibility for subsequent DE meetings. This automated approach not only streamlines the process but also supports better alignment and collaboration within the clinical team and with stakeholders.

Despite the clear benefits, we acknowledge the need for further refinement before a wider roll-out. Future steps include additional pilots to refine the template, integrating improvements into the `autoslideR` and `autoslider.core` packages, and enhancing documentation to support broader adoption. Our ongoing efforts aim to provide a robust and scalable tool that facilitates reliable and efficient DE decision-making across various studies.

REFERENCES

1. https://cran.r-project.org/web/packages/autoslider.core/index.html
2. https://insightsengineering.github.io/nest/
3. https://cran.r-project.org/web/packages/officer/index.html
4. https://cran.r-project.org/web/packages/rtables/index.html

ACKNOWLEDGMENTS

We would like to thank the `autoslideR` development team (including but not limited to Joe Zhu, Yinqi Zhao, Xiaoli Duan), the Roche/Genentech Dose Escalation Working Group, the pilot study clinical and safety teams, as well as Data & Statistical Sciences (DSS) management support.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:

Jasmina Uzunovic
Hoffman-La Roche Canada
jasmina.uzunovic@roche.com

Aditi Qamra
Hoffman-La Roche Canada
aditi.qamra@roche.com

Nina Qi
Roche/Genentech
qi.ting@gene.com