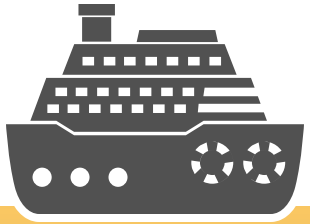
ML11: Leveraging AI for Seamless SAS to R Migration

PHUSE EU Connect 2025



Challenges and Opportunities

Agenda & Speaker Intro

Why are we moving?

What are we moving?

Where are we moving?

How are we moving → With AI ☺

Closing remarks

Corinna Stöckinger (Presenter)

Data Science Programmer & Leader of the Experimental Medicine department's SAS2R task force

corinna.stoeckinger@boehringer-Ingelheim.com

Boehringer Ingelheim

Milena Kraus (Presenter)

Software & Migration Engineer with 15+ years of experience in R/Pharma

milena.kraus@analytical-software.de

HMS

Robert Bauer (Co-Author)

Senior Data Scientist & Lead Expert for AI-assisted code migration

robert.bauer@analytical-software.de

HMS

Leveraging AI for Seamless SAS to R Migration

Why are we moving?

What are we moving?

Where are we moving?

How are we moving → With AI ☺

Closing remarks

Why are we moving?

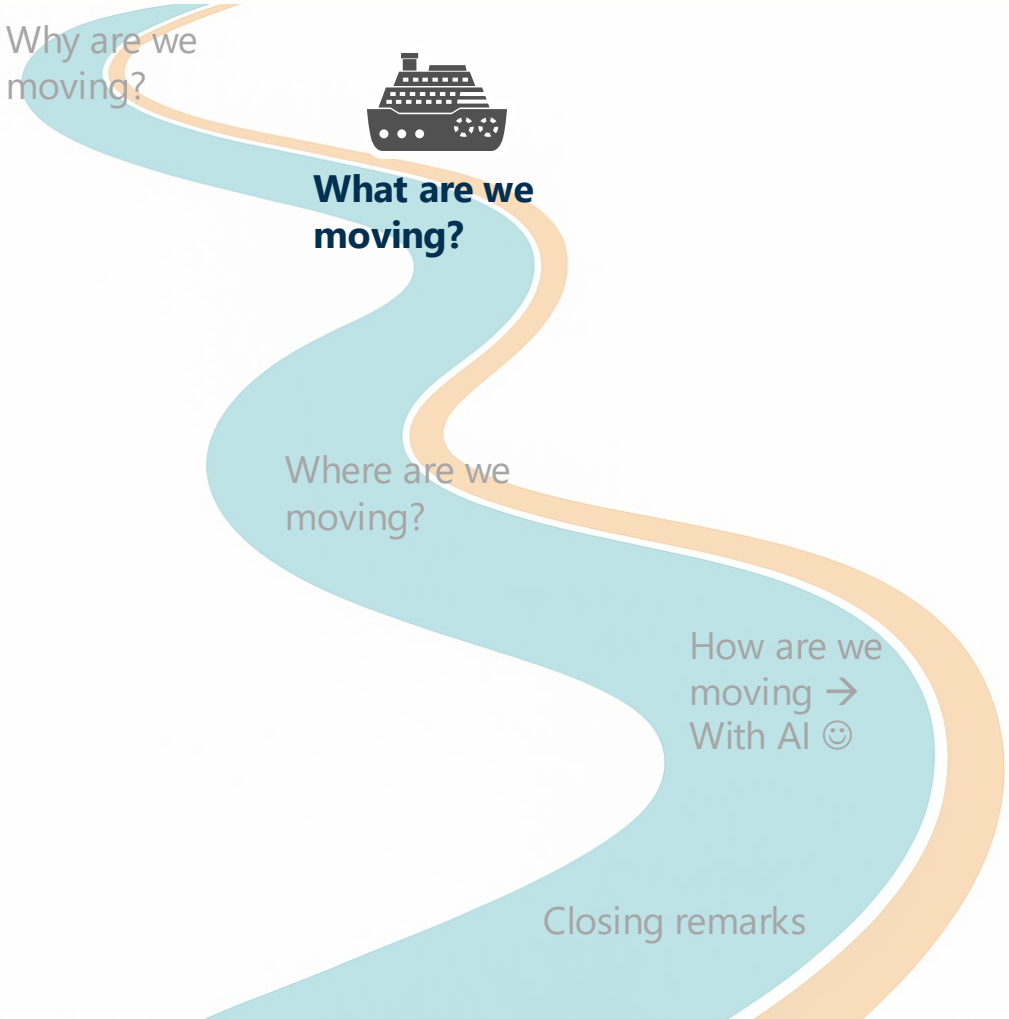
Open-source =

Arguments for the migration from SAS to R

- Transparency
- Financial reasons
- Software Unification
- Flexibility and Customization
- AI Integration
- Collaboration and Innovation
- Interoperability
- Streamlined processes

HMS Boehringer Ingelheim

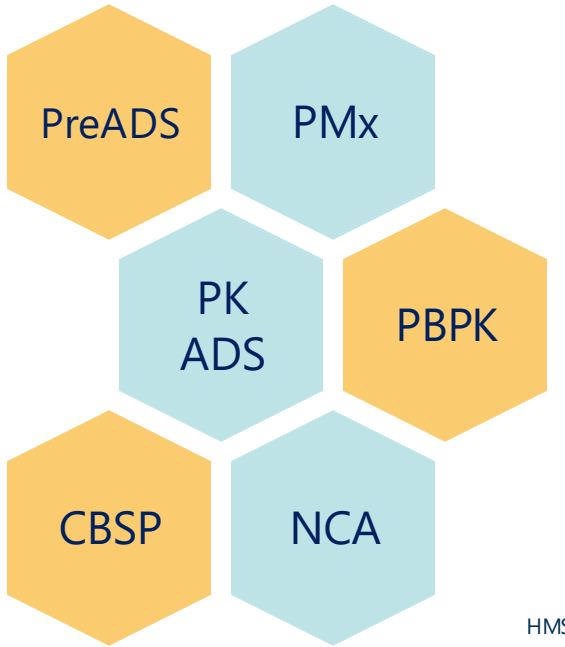
Leveraging AI for Seamless SAS to R Migration



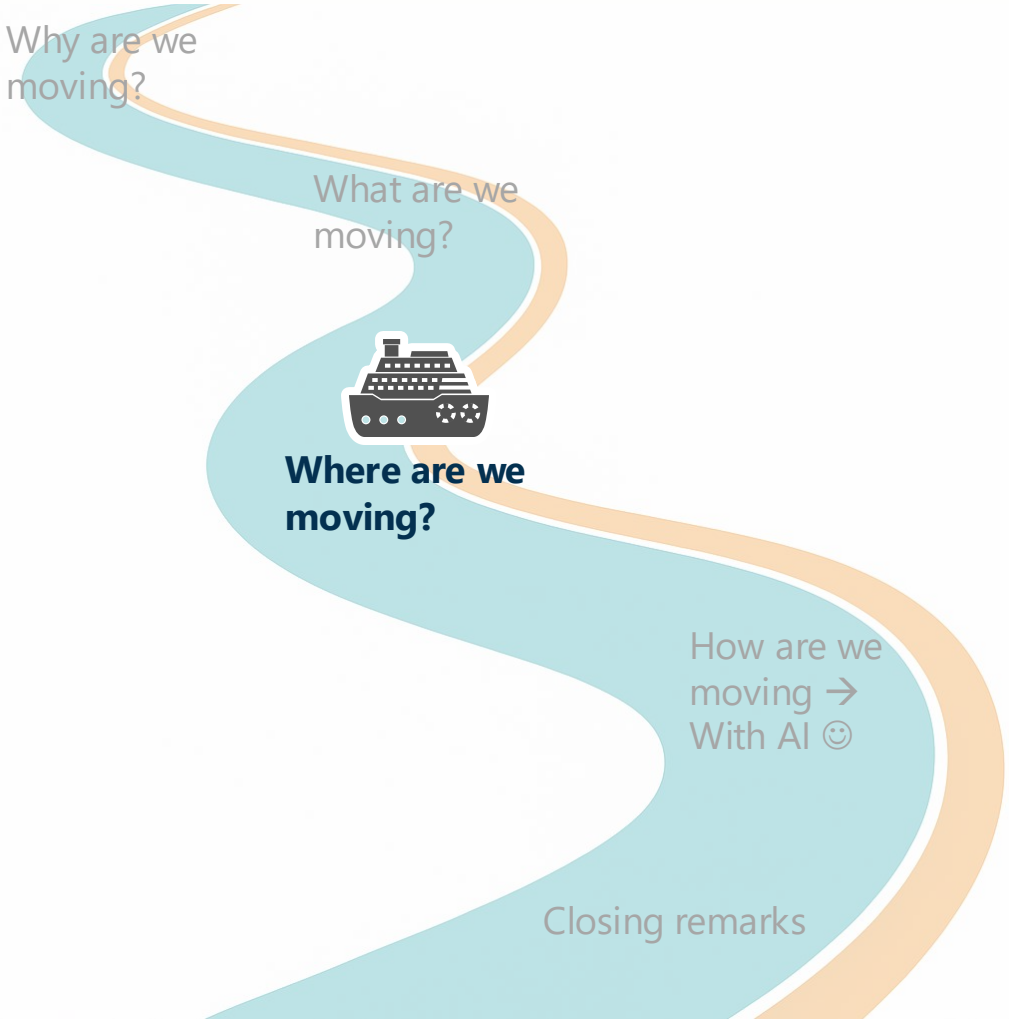
What are we moving?

Boehringer PK/PD SAS Code Base

- 20,000 lines of code
- SAS Macros & Templates
- **PreADS**: Preparation of the relevant SDTM domains, e.g. date transformations, derivation of baseline values of relevant covariates
- **PMx**: Creation of popPK analysis datasets (similar to CDISC ADPPK standard, following NONMEM structure)
- **PK ADS**: Creation of analysis datasets for pharmacokinetic, pharmacodynamic and biomarker data (similar to CDISC ADNCA standard), calculation of NCA parameters (similar to CDISC PP standard), creation of analysis datasets for immunogenicity data (ADIS).
- **PBPK**: Creation of analysis datasets for physiologically based PK (PBPK) analyses (based on PK ADS)
- **CBSP** (Clinical Bioinformatics and Systems Pharmacology): Creation of analysis datasets for gene and protein expression analyses



Leveraging AI for Seamless SAS to R Migration



Where are we moving?

Infrastructure

- Software agnostic statistical computing environment RISE (Report Integrated Statistical Environment)
- Custom-developed & cloud-based
- Functions:
  - Program analysis macros and formally validate them
  - Organize the results into a report form
  - Generate PDF documentation and reports for submission to regulatory agencies
- SAS Studio on SAS Viya
- RStudio Pro on a Posit workbench

See: PHUSE24-AD05: Development of a Customised Statistical Computing Environment for Clinical Trial Data Analysis

R Frameworks

R Base

› Procedural syntax similar to SAS

Tidyverse

› Collection of R packages for data manipulation, e.g. dplyr, tidyr, stringr, and readr

› Improves readability and modularity, but requires some upskilling

Pharmaverse

› Ecosystem of R packages tailored for creation of CDISC compliant ADaM datasets and clinical trial reporting, e.g. admiral

› Templates and user guides (e.g. workflow for creating ADNCA, ADPPK)

Framework Choice Matters

R Base

```
data$IsDuplicate <- duplicated(data[sortkey])
```

- One line of code
- Assumes proper data/matching index

Tidyverse

```
data %>%
        dplyr::distinct(sortkey, keep_all = TRUE)
```

OR

```
data %>%
        dplyr::arrange(dplyr::across(dplyr::all_of(sortkey))) %>%
        dplyr::group_by(dplyr::across(dplyr::all_of(sortkey))) %>%
        dplyr::mutate(rn = dplyr::row_number()) %>%
        dplyr::mutate(isDuplicate = dplyr::case_when(rn > 1 ~ 1, .default = NA)) %>%
        dplyr::select(-rn) %>%
        dplyr::ungroup()
```

- Tidyverse assumes duplicates should be removed and recommends to use distinct()
- Creating a duplicate flag variable requires helper structures

Admiral

```
signal_duplicate_records(adsl, sortkey, cnd_type = "warning")
#> Warning: Dataset contains duplicate records with respect to `USUBJID`
#> ℹ Run `admiral::get_duplicates_dataset()` to access the duplicate records
```

- Admiral expects duplicates to be erroneous data and throws an exception
- Provides helpers to locate duplicates

Migration vs. Refactoring

Tidyverse/R Base

- Follows existing SAS logic
- Uses tidyverse/R base as applicable
- „Migration“

VS

Admiral

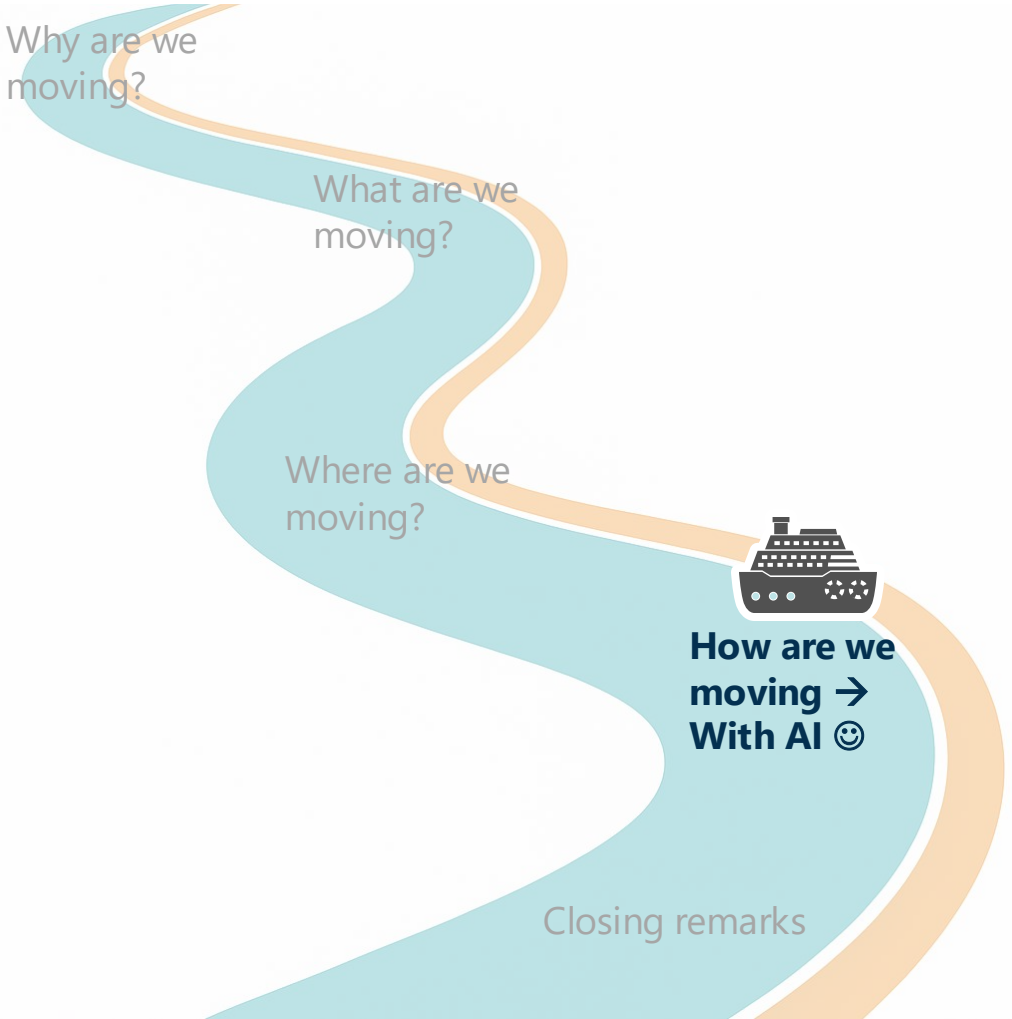
- Follow admiral guide for ADSL
- Strict tidy design principle
- „Refactoring“

For our project:
Migration + Admiral for (currently few) selected parts

	Tidyverse/R Base	Admiral
Implementation effort	Medium	High
Similarity to existing SAS code base	High	Low
Inherent refactoring needs	Medium	High
Upskilling effort	Low	Medium
Open-source community benefits	Low	High
Debugging output/program insights	Easy	Possible
Templates/customization	Easy	Medium
Future perspectives/cross industry	-	Potential given
Adherence to CDISC standard	Not enforced	High

HMS Boehringer Ingelheim

Leveraging AI for Seamless SAS to R Migration



How are we moving?

Use AI for Translation

LLMs without proper guardrails

- Context overflow
- Trained for code generation/lack of SAS
- Incorrect handling of complex SAS idioms
- Oversimplification of critical functionality
- Lack of uniform style, naming, error handling, and data type conventions
- Time consuming manual iterations

Use AI for Translation

LLMs without proper guardrails

- Context overflow
- Trained for code generation/lack of SAS
- Incorrect handling of complex SAS idioms
- Oversimplification of critical functionality
- Lack of uniform style, naming, error handling, and data type conventions
- Time consuming manual iterations

Our Approach

We couple GenAI with a **controlled process** and **human-operated guardrails**

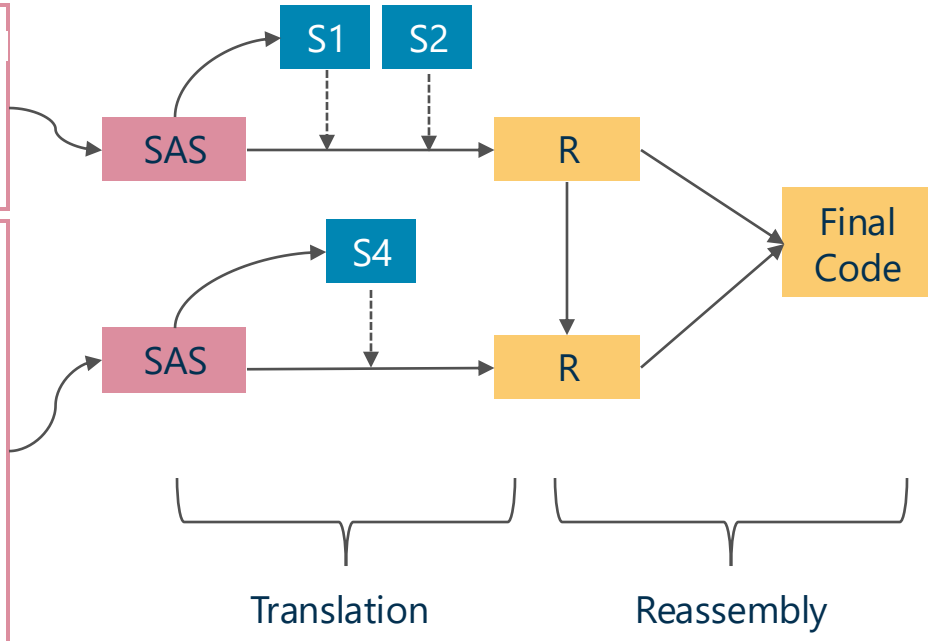
- Decomposition & Reassembly
- Migration Strategies
- Automation

Decomposition & Reassembly

- Split complex, monolithic input (macro, larger DATA steps) into smaller, semantically coherent chunks
- Decomposition requires reassembly
- Why prefer small chunks?
  - Reduced prompt noise and ambiguity
  - Improved determinism → similar problems are solved in similar ways

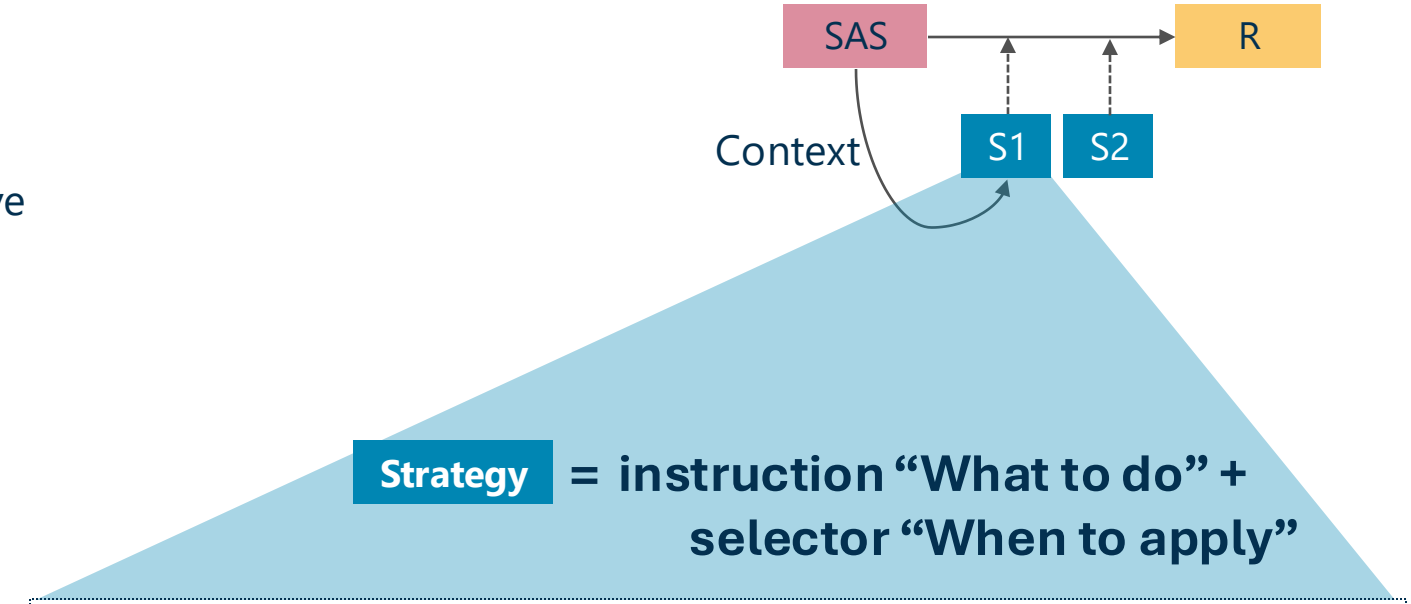
```
%macro pmx_read(in=, out=, vars=_ALL_, wkey=1 eq 1, skey=_ALL_,
  create_ptno=Y, debug=N, usubjidsep=%str(-));

    data __tmpin(where = (&wkey));
       set &in;
    run;
    data __tmpin(keep = &vars);
       set __tmpin;
    run;

  /* check for duplicates */
  %pmx_dupobs(in=__tmpinx, out=__tmpoutx, sortkey=&skey, debug=&debug)

  data &out;
     set __tmpoutx;
     by &skey;
```

```
%macro pmx_dupobs(in=, out=, sortkey=, dupflg=dupl, debug=N);
   %if "&out" = "" %then %do;
      %put ERROR: Please specify a dataset name in input parameter OUT. M
      %goto exit;
   %end;

   data __tmpoutdupobs;
      set &in;
   run;

   /* perform sort to get duplicates */
   proc sort data = __tmpoutdupobs nodupkey dupout = __dupstmp;
      by &sortkey;
   run;
   data __dupstmp;
      set __dupstmp;
         &dupflg = 1;
   run;
   data __tmpoutdupobs;
      set __tmpoutdupobs __dupstmp;
   run;

   /* third - sort dups in again, in correct order, to get whole dataset
   proc sort data = __tmpoutdupobs out = &out;
      by &sortkey;
   run;

   %if &debug = Y %then %do;
   %end;
   %else %do;
      proc delete data = __dupstmp __tmpoutdupobs;
      run;
   %end;
```



Migration Strategies

- System prompt is programmatically enriched with “strategies” to improve translation
- Context sensitive → Only relevant strategies are selected
- Can be very simple and generic

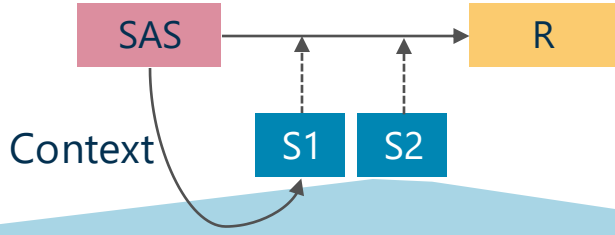



```
TEMPLATE = """
<strategy_general_coding_remarks>
 Some general rules:
 - Use tidyverse functions where possible.
 - Use package::function() syntax for all non base R functions.
 - For all variables with leading underscores replace the underscore with the prefix "tmp_".
 - For all variables containing a path put "_path" as a suffix, when initializing and using them even
 - Dataframe names are all lowercase, i.e., PREdf_c becomes predf_c, NCAres becomes ncares, etc.
 - Use readxl and writexl for reading and writing Excel files.
 - Only return plain R code inside <r_code>...</r_code> tags.
 - DO NOT ADD ADDITIONAL COMMENTS OR INSTRUCTIONS TO YOUR RESPONSE!
</strategy_general_coding_remarks>
"""  # noqa

def selection_function(ast_model: AST_MODELS.ASTModelGeneric, future_state: dict[str, Any]) -> bool:
    return True
```

Migration Strategies

- System prompt is programmatically enriched with “strategies” to improve translation
- Context sensitive → Only relevant strategies are selected

- Can be very simple and generic or
- Complex and specific
- Lengthy strategies should not clutter the prompt if they are not needed!

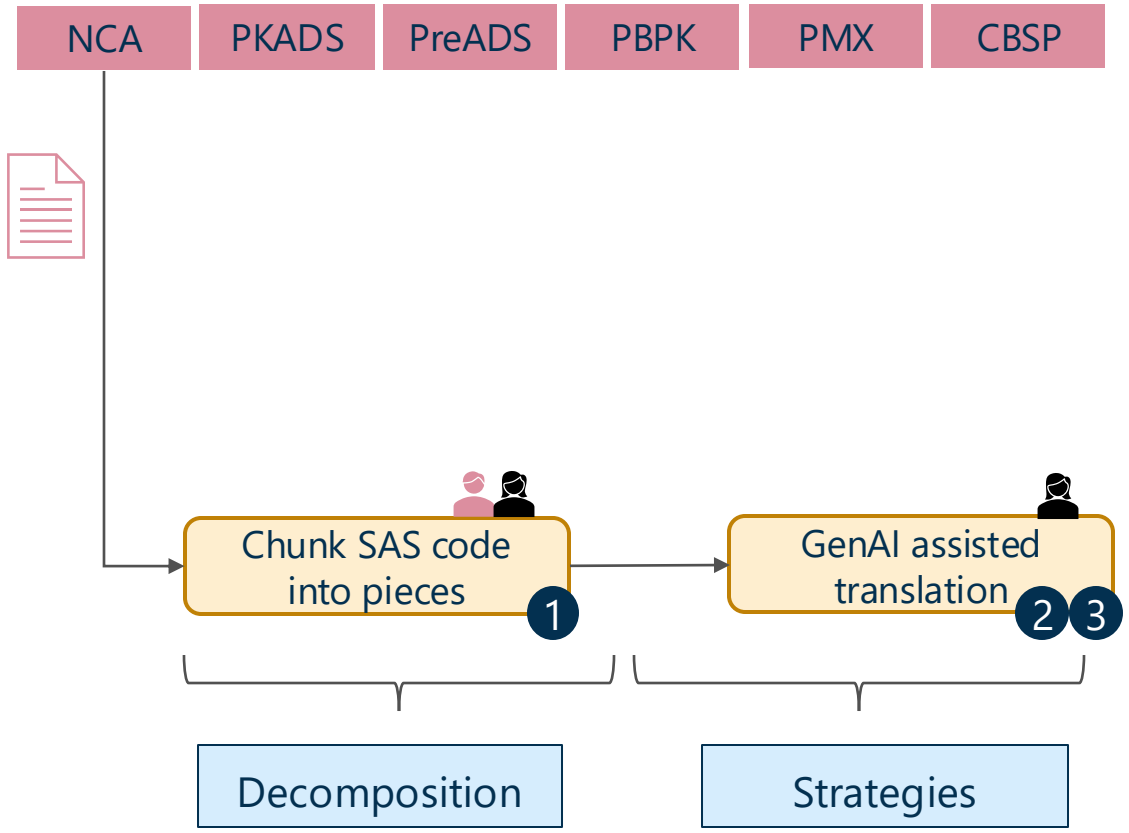



```
TEMPLATE = """
<strategy_format_missing_values>
This strategy defines how to handle missing values (NAs).

After loading, convert blank strings to `NA` using `admiral::convert_blanks_to_na()`.
After that, assume NA's are not typed and ignore explicit formatting from SAS.
Typed NA's are not needed in R and will complicate further processing.

Example of NA conversion as it's done in admiral:
<r>
ds <- haven::read_xpt(rawdata_path) |> admiral::convert_blanks_to_na()
</r>

Example of not typing R NA'S despite SAS NA's always being typed:

<r>
data <- dplyr::mutate(data, VAR1 = NA) # Note how this is NOT NA_real_ or NA_character_
data %>% dplyr::filter(!is.na(VAR1)) # Note how this can be filtered with is.na() instead
</r>

<sas>
if compress ( lowcase ( studyid ) ) in ( "" " " 'n.a." "missing" "-" "." ) ....
</sas>
<r>
dplyr::if_else(!is.na(STUDYID), ... , NA)
</r>

<explanation>
ALWAYS use untyped NA's in R. NEVER use typed NA's like NA_real_ or NA_character_.
Note how the filter and if_else are implemented with is.na() instead of all possible SAS N
</explanation>
```

HMS Boehringer Ingelheim

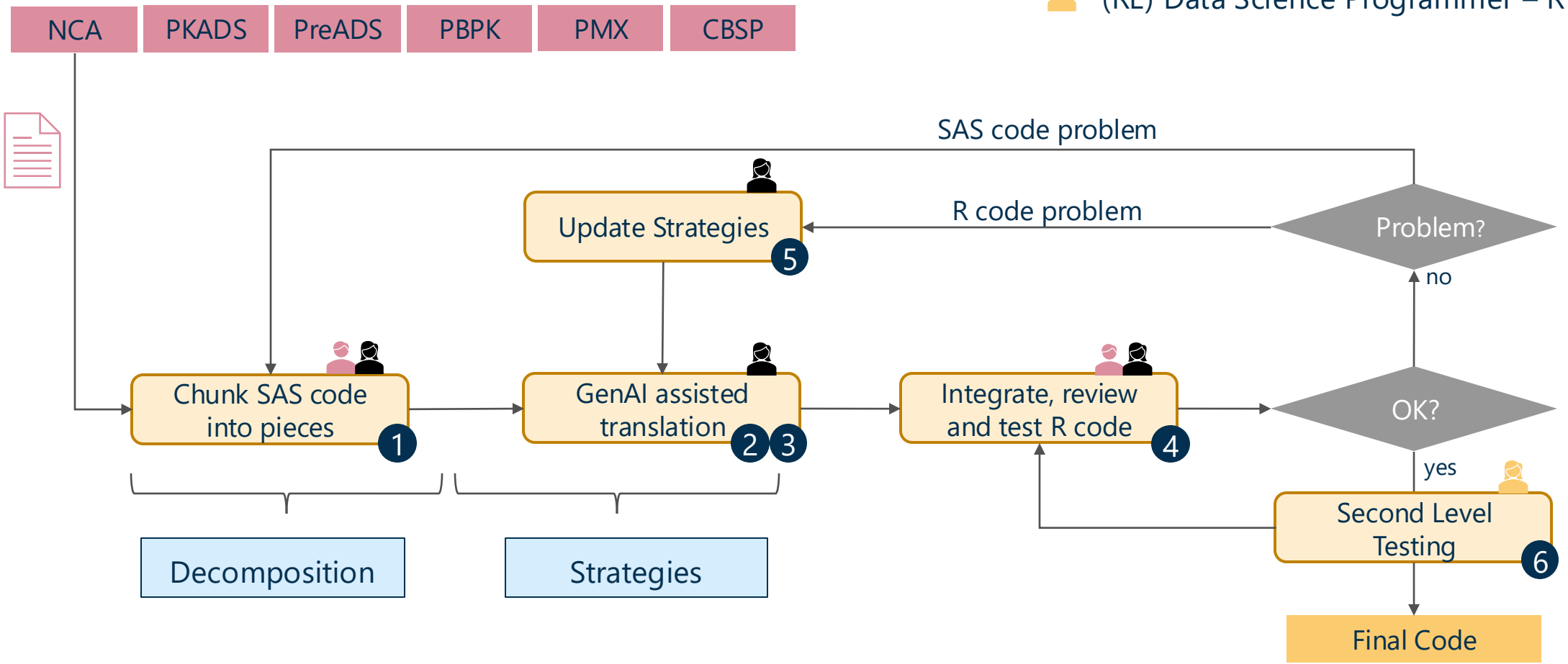
Exploratory Migration Approach

- (SE) Data Science Programmer – SAS
- (ME) Migration Engineer – SAS & R
- (RE) Data Science Programmer – R



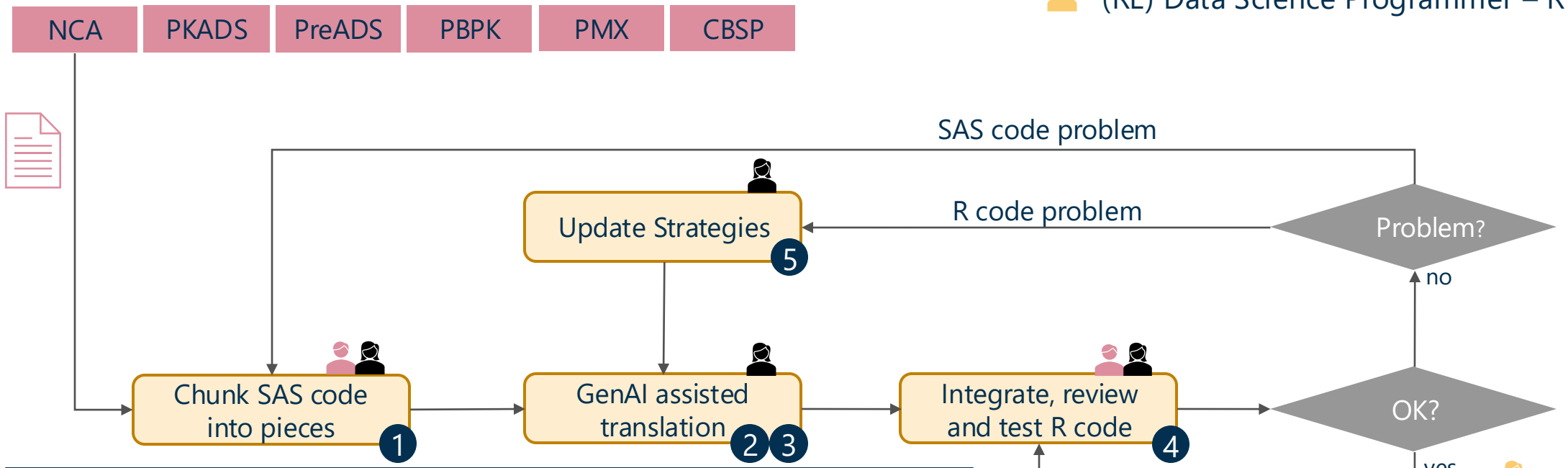
Exploratory Migration Approach

(SE) Data Science Programmer – SAS
(ME) Migration Engineer – SAS & R
(RE) Data Science Programmer – R



Exploratory Migration Approach

(SE) Data Science Programmer – SAS
(ME) Migration Engineer – SAS & R
(RE) Data Science Programmer – R



- Posit Workbench Pro
- Ellmer R package and Boehringers LLM endpoints (claude sonnet 4)
- Custom R utilities for iterative tasks, authentication and testi
- Manual work, copy/paste of code and prompts/strategies

ellmer

HMS Boehringer Ingelheim

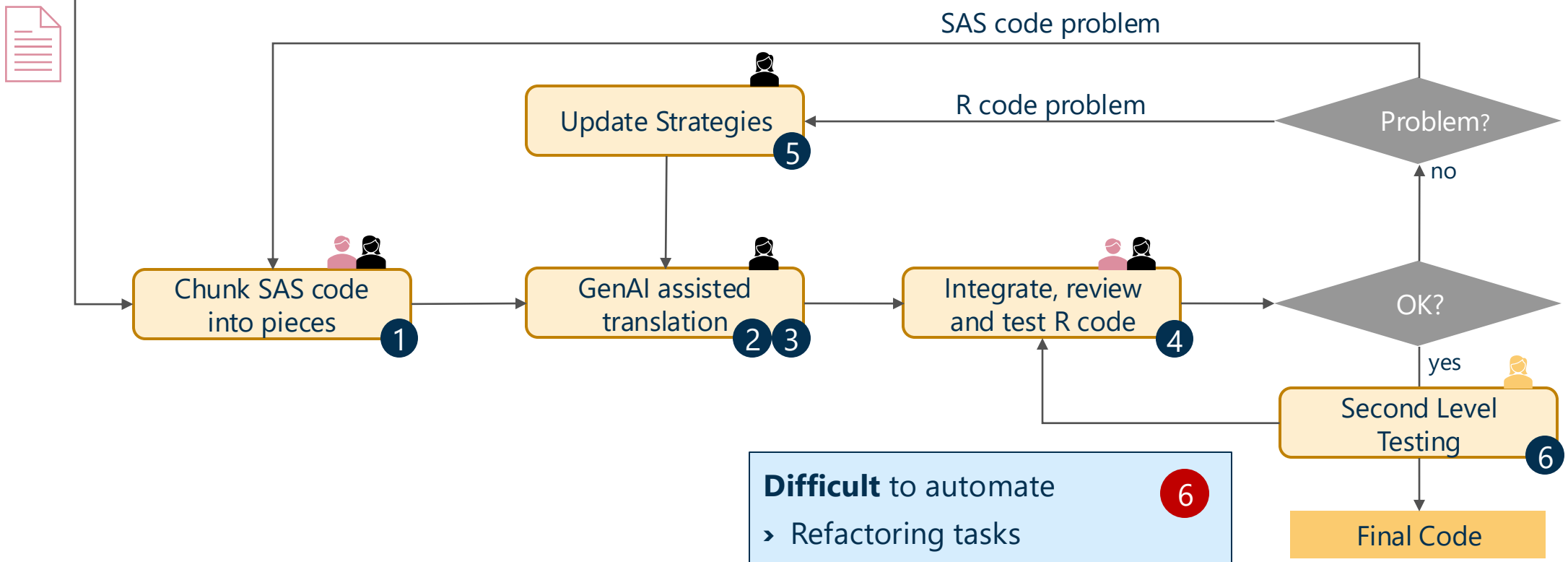
Path to Automation

Room for **improvement/automation**

- Code decomposition
- Strategy definition, selection and refinement
- (Batch) translation
- Feedback loops and testing

1 2 3 4 5

NCA | PKADS | PreADS | PBPK | PMX | CBSP



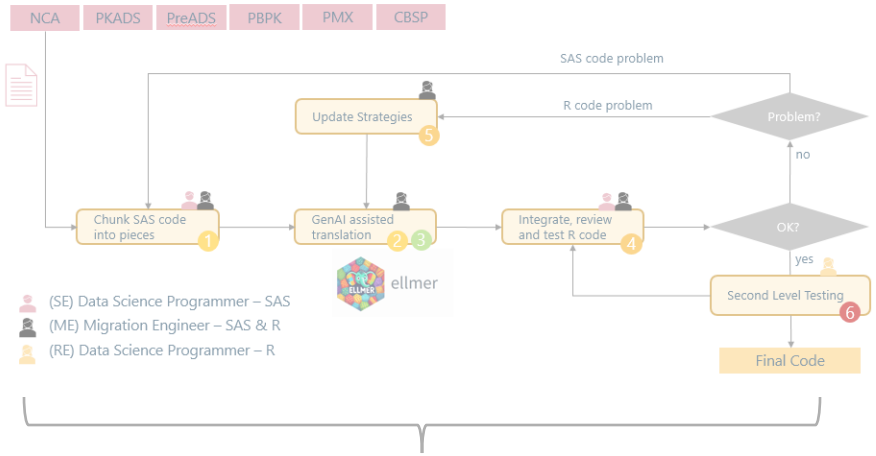
Difficult to automate

6

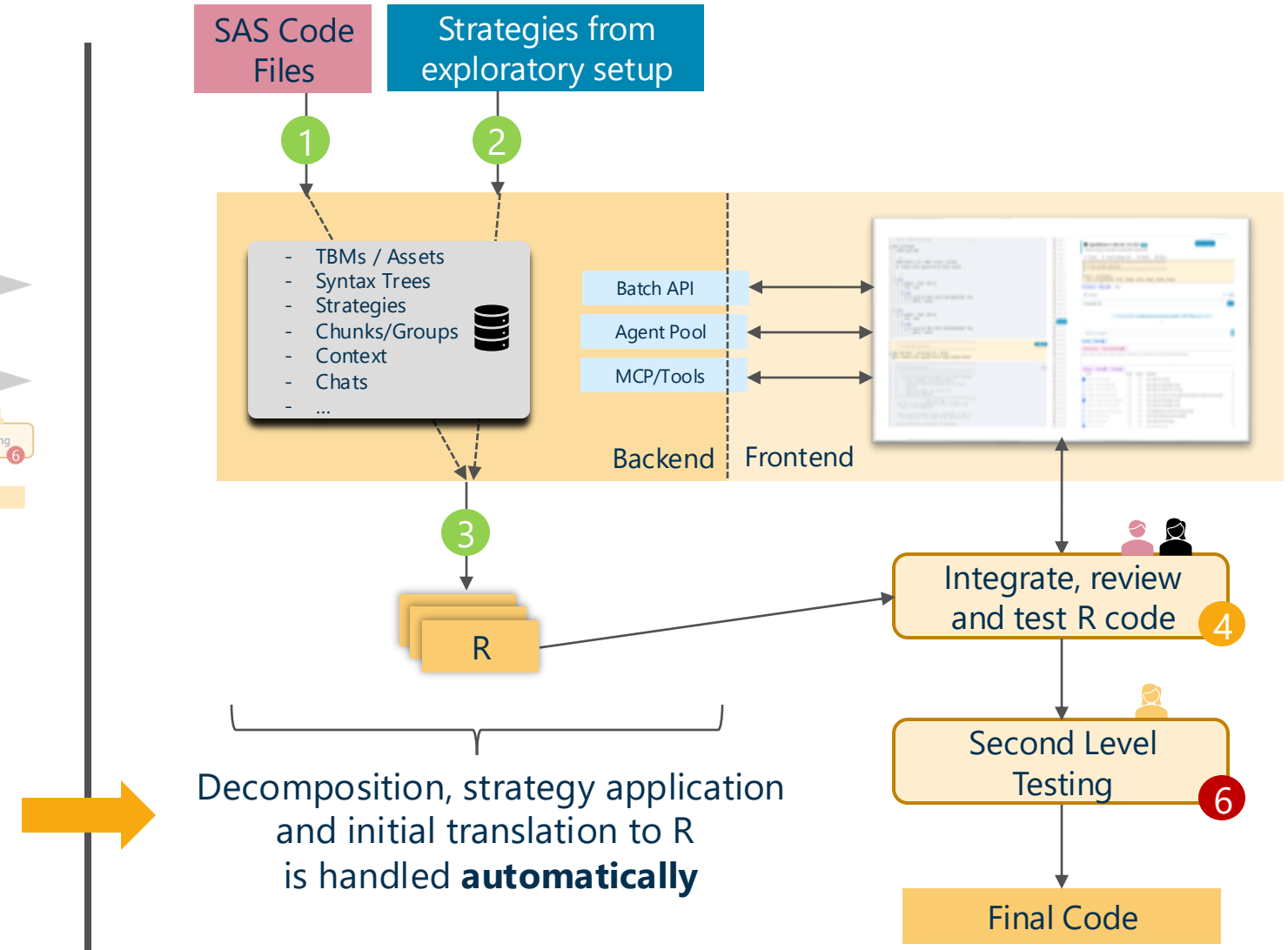
- Refactoring tasks
- Domain knowledge issues

Path to Automation

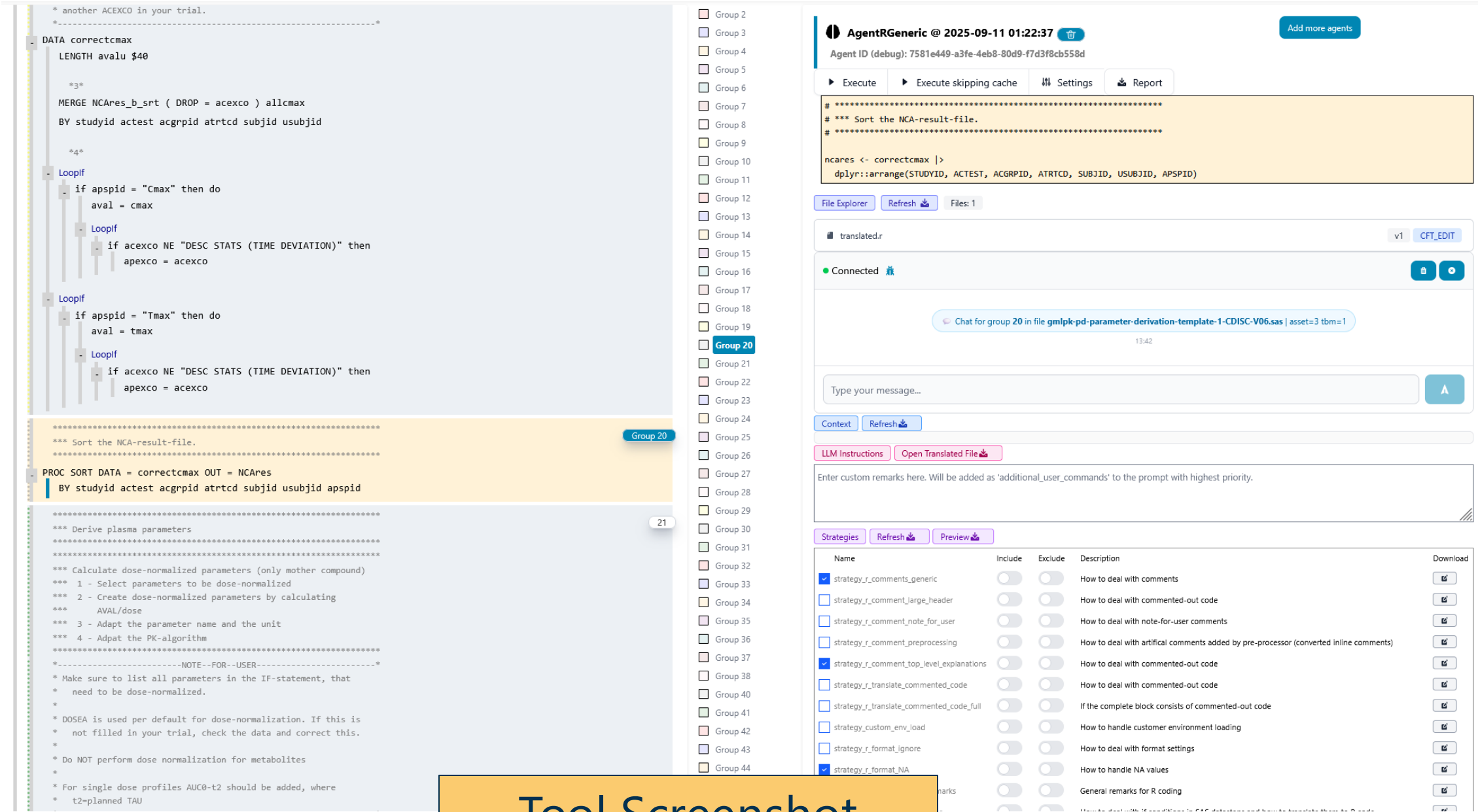
Exploratory Approach



The whole process is **handled manually** by the ME, SE and RE

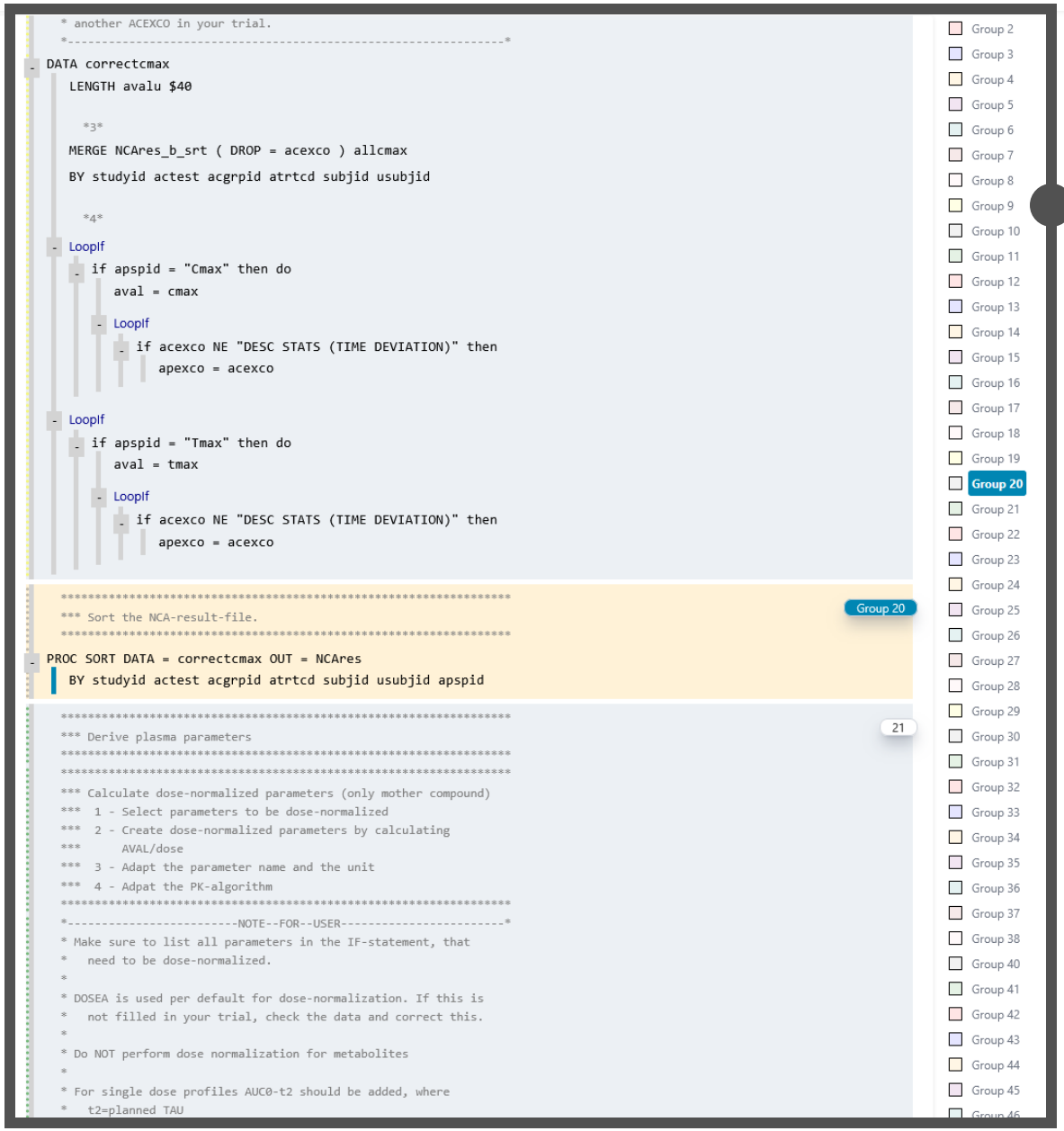


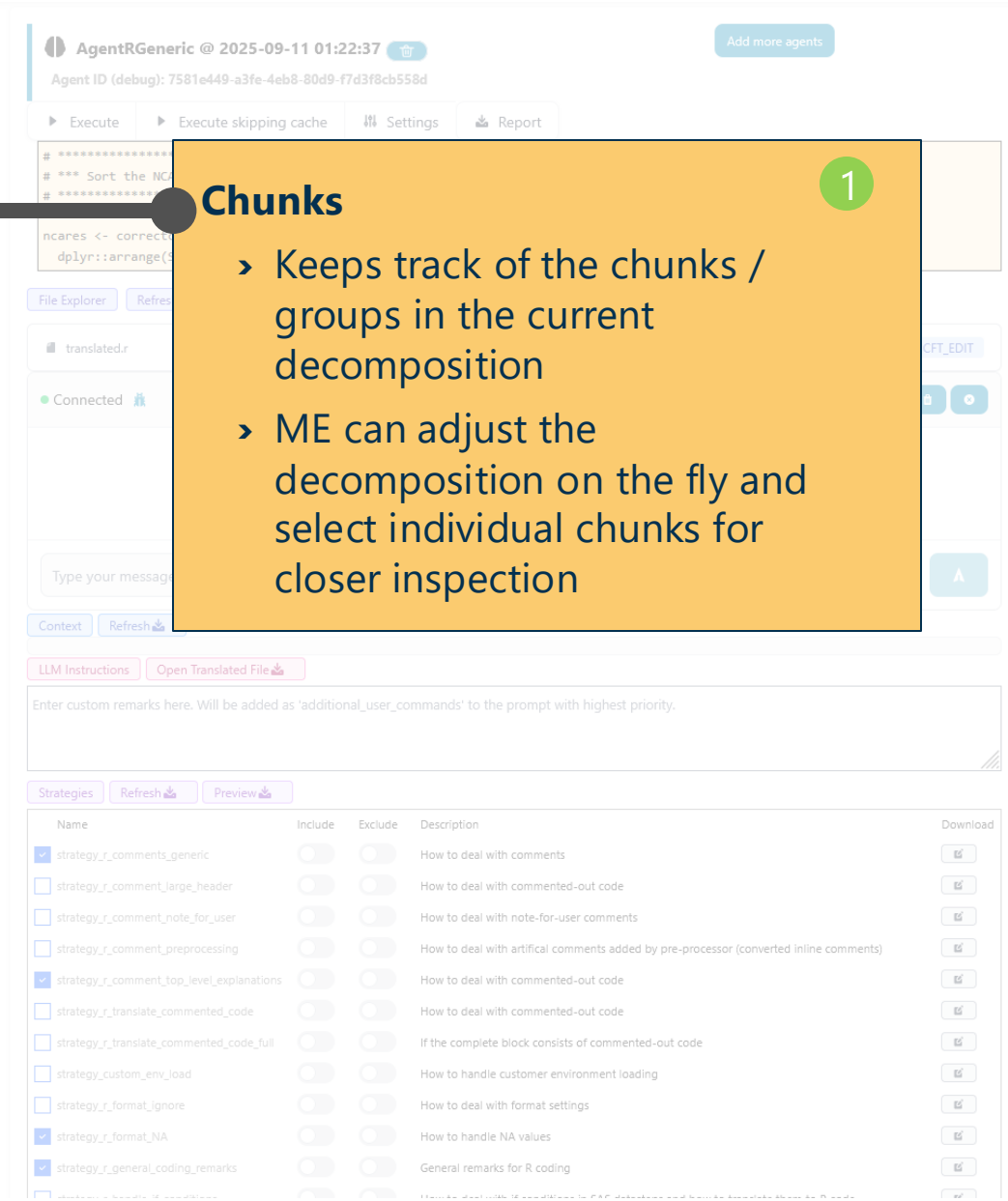
Decomposition, strategy application and initial translation to R is handled **automatically**



Tool Screenshot

Disclaimer: The tool shown is not a commercially available product. It is developed in-house and used by HMS migration engineers.





Chunks

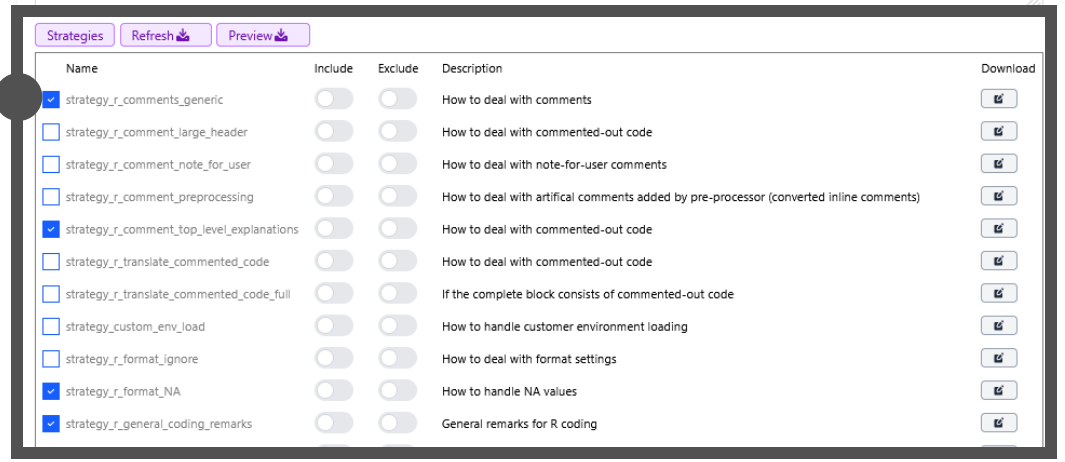
1

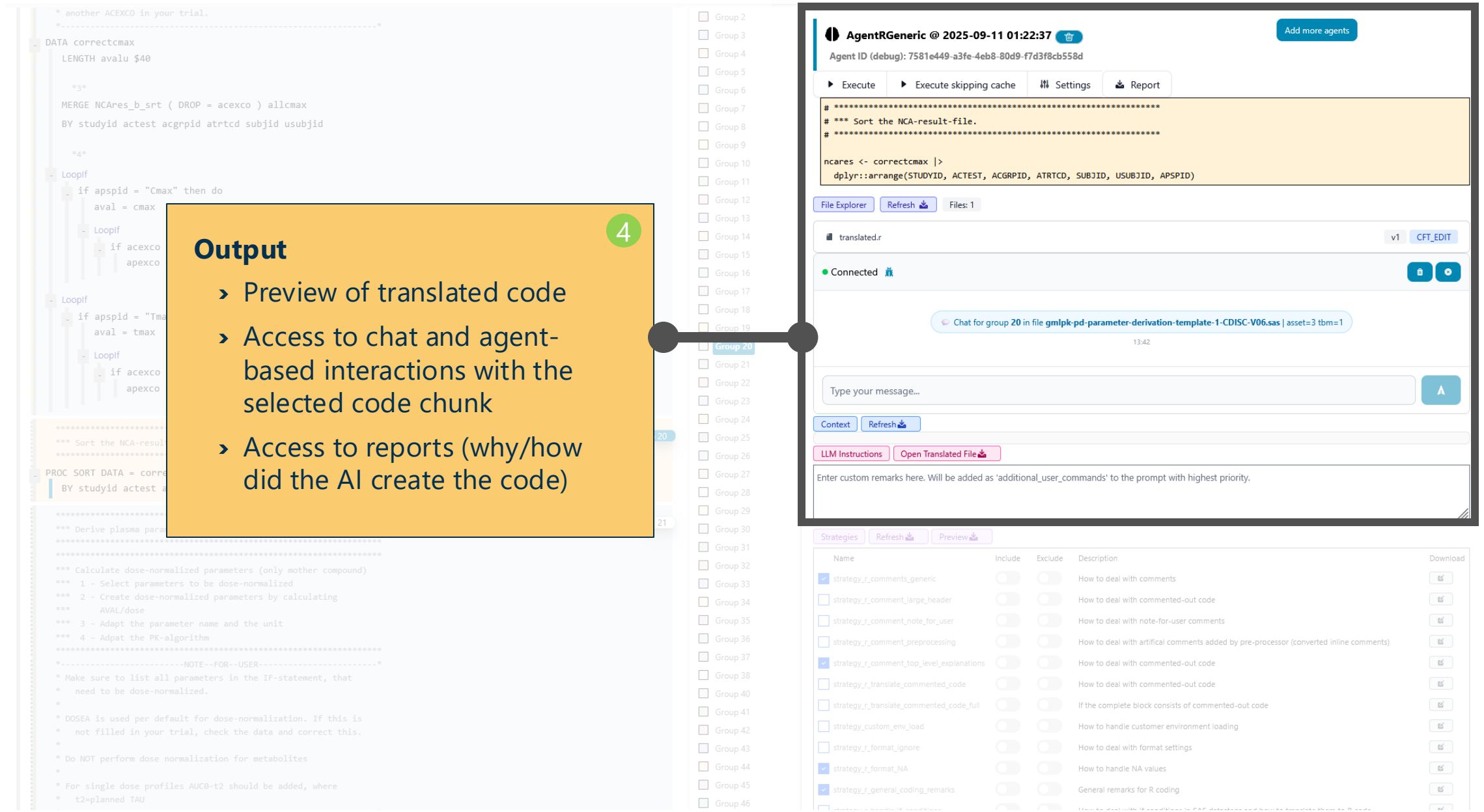
- Keeps track of the chunks / groups in the current decomposition
- ME can adjust the decomposition on the fly and select individual chunks for closer inspection

Strategies

2

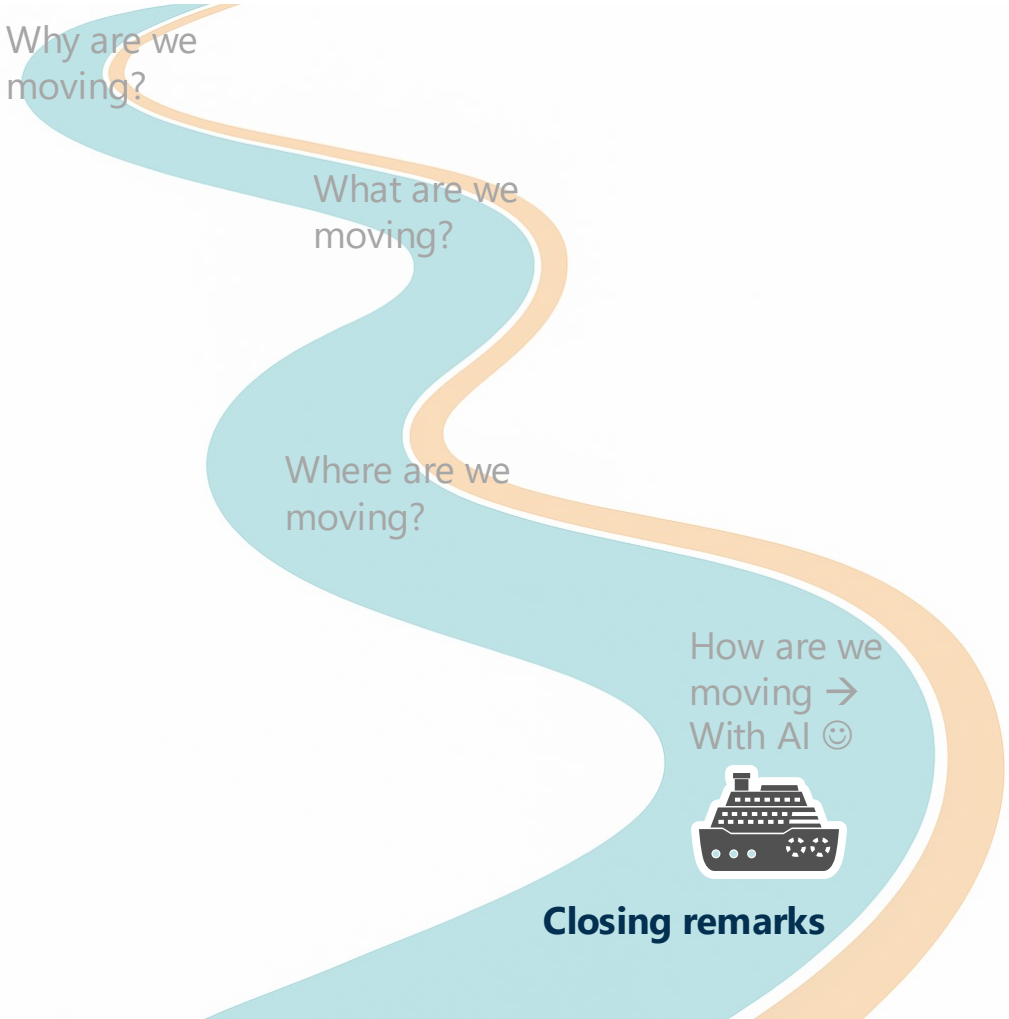
- Shows available strategies and whether they are applied to the selected chuck
- ME can edit the strategies
- ME can overwrite decisions made by the system





Disclaimer: The tool shown is not a commercially available product. It is developed in-house and used by HMS migration engineers.

Leveraging AI for Seamless SAS to R Migration



Closing Remarks

Closing Remarks

Challenges

- Target frameworks need careful consideration
- LLM guardrails crucial for efficiency and consistency
- Domain complexity requires a lot of manual work and expertise

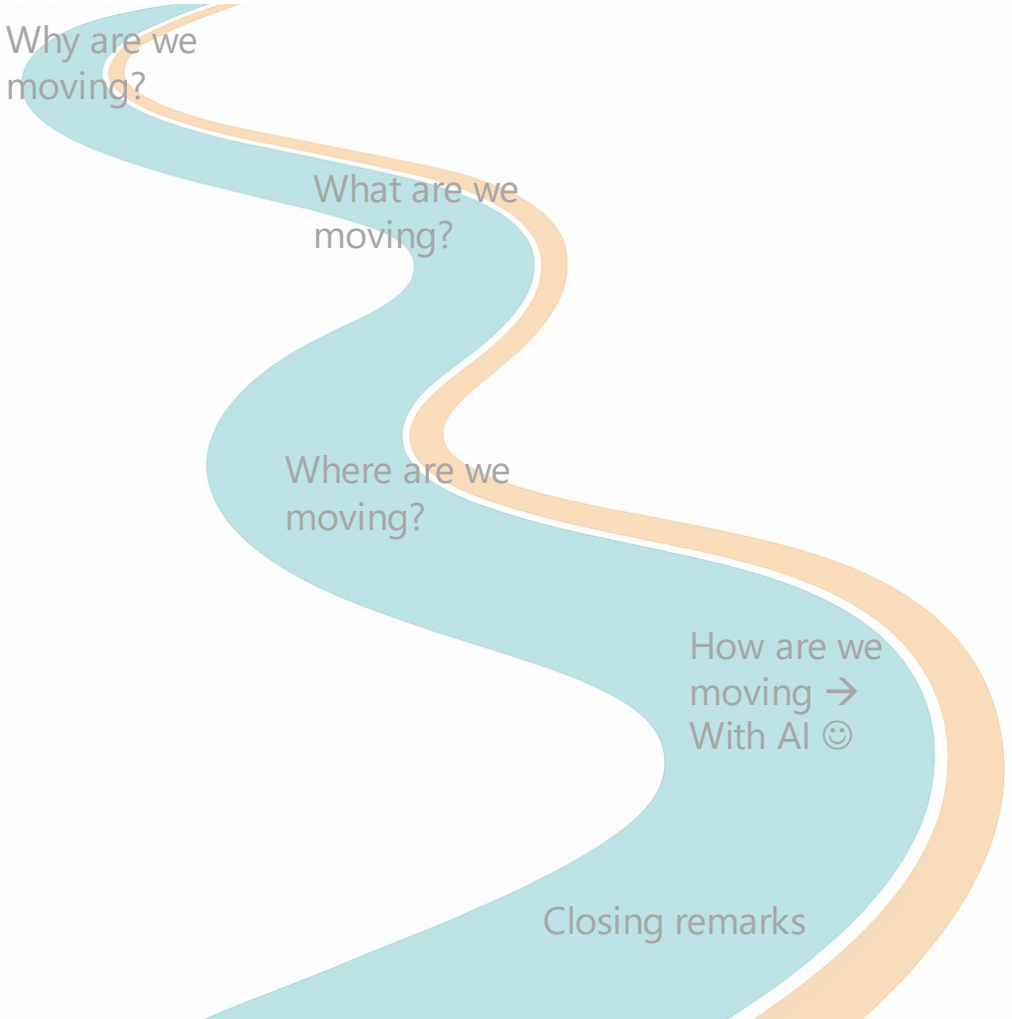
Opportunities

- Leveraging R's flexibility and open-source packages
- Tidy data principles and functional code design
- Lots of opportunities for automation

Future Directions

- After completion of translation: Integration into RISE environment
- User enablement
- Path to full automation: Direct code execution, automated testing, and intelligent migration strategies

Leveraging AI for Seamless SAS to R Migration



Thank you!