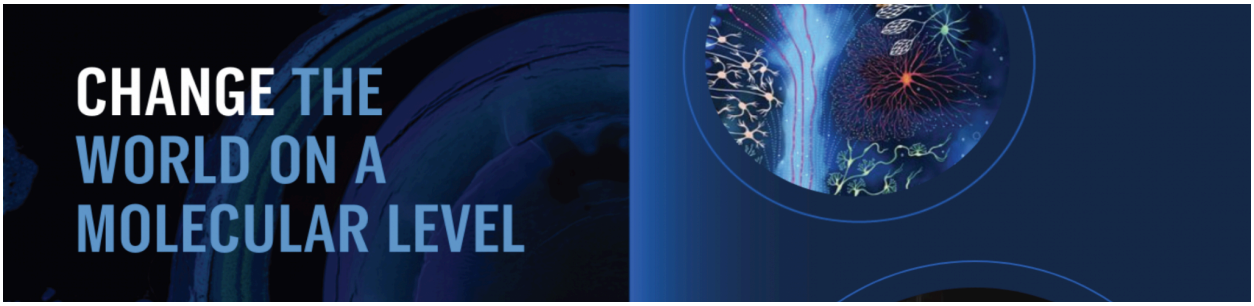


Analytical Data Science Programmer Coding Assessment

Introduction

Welcome to the Pharmaverse Expertise and Python Coding Assessment. This assessment is designed to evaluate your skills and proficiency in the following areas:

- **Pharmaverse Ecosystem**: Working with open-source packages designed for clinical trial data standards, including SDTM (Study Data Tabulation Model) and ADaM (Analysis Data Model).
- **Data Manipulation & Derivations**: Utilizing modern R tools like {dplyr}, {tidyr}, and Pharmaverse packages such as {admiral}, {sdtm.oak}, and {gtsummary}.
- **Clinical Reporting**: Generating Tables, Listings, and Graphs (TLGs) for regulatory submissions
- **Python:** Applying Python for data science
- **Clean Code Practices**: Writing reproducible, efficient, and well-documented R code
- **Problem-Solving & Adaptability**: Demonstrating the ability to quickly learn new tools and apply them effectively to solve challenges.
- **Efficient use of AI:** Leverage modern AI tools to accelerate learning new concepts and complete coding exercises.

The assessment includes practical coding exercises focused on clinical trial reporting in the pharmaceutical industry and core software development. It is designed to evaluate your technical expertise, critical thinking, and approach to software design. Candidates are allowed to use AI-enabled coding assistants.

Assessment Format

Questions

This coding assessment consists of **4 questions**:

- **Questions 1-3 [Required]**: SDTM, ADaM, and table creation using open source R packages from Pharmaverse
- **Questions 4 [Bonus]**: Python question

What to Submit

1. **GitHub Repository Link**
    - Create a public GitHub repository for your solutions
    - Repository should be well-organized with clear folder structure (one folder for each question)
    - Include a comprehensive README.md explaining the repo structure and the contents of each folder that could be helpful for the reviewer.

2. **Code Files**
    - Each question should have a dedicated folder and its own file (e.g., `question_1.R`, `question_2.R`, etc.)
    - Include comments explaining key logic, especially for derivations

3. **Video Explanation (2 minutes)**
    - Record a brief screen-share walkthrough of your repository
    - Explain your approach, design decisions, and key challenges overcome
    - Discuss what you learned from the exercise

Timeline

- **Submission Deadline**: 1 week
- **Submission Format**: GitHub repository link and video link submitted as part of the Phenom interview
- **Interview Scheduling**: Selected candidates will be contacted for follow-up discussion

Evaluation Process

Your submission will be evaluated on:

- **Code Quality**: Clarity, efficiency, and adherence to R best practices
- **Correctness**: Does the code produce the expected output?
- **Documentation**: Are functions and code well-documented?
- **Problem-Solving**: How creative and thorough are your solutions?
- **Communication**: Can you explain your approach clearly?

Posit Cloud

Candidates can use a Posit Cloud free plan or any other plan as mentioned in the Posit website (https://posit.cloud/plans) to work on R related questions. Once logged in, create a new workspace to clone your Github repository to get started.

R Version

R 4.2.0 and above.

Required R Packages

The necessary packages, along with all dependencies, can be installed from CRAN using the provided command. Please note this is only an example; additional R packages will be needed to complete the assessment.

```
None
install.packages(c("admiral", "sdtm.oak", "gt", "ggplot2"))
```

Learning Resources

Each question has question specific learning resources and a hint to answer the question. Also, here is the consolidated list of all learning resources:

Pharmaverse & CDISC

- **Pharmaverse Examples**: https://pharmaverse.github.io/examples/
- **{admiral} Documentation**: https://pharmaverse.github.io/admiral/
- **{sdtm.oak} Documentation**: https://pharmaverse.github.io/sdtm.oak/
- **CDISC Standards**:
    - SDTM IG: https://www.cdisc.org/standards/foundational/sdtmig
    - ADaM IG: https://www.cdisc.org/standards/foundational/adam
- **Coursera**: Hands On Clinical Reporting Using R
- **Pharmaverse site:** https://pharmaverse.org/

Data Manipulation & Visualization

- **dplyr Documentation**: https://dplyr.tidyverse.org/
- **tidyr Documentation**: https://tidyr.tidyverse.org/
- **ggplot2 Documentation**: https://ggplot2.tidyverse.org/
- **gt (Great Tables)**: https://gt.rstudio.com/

Testing & Best Practices

- **testthat Package**: https://testthat.r-lib.org/
- **R Style Guide**: https://style.tidyverse.org/
- **Advanced R (OOP & Functional Programming)**: https://adv-r.hadley.nz/

Additional Resources

- **Pharmaverse Blog**: Articles on Pharmaverse packages and clinical trial data programming https://pharmaverse.github.io/blog/
- **YouTube**: Search for "admiral R package tutorial" or "SDTM programming in R using {sdtm.oak} package" to view the videos in “R in Pharma” channel. There are many more videos in R in Pharma or R Consortium youtube channels.
- **RStudio Community**: https://community.rstudio.com/ (Q&A forum)

Pharmaverse

Learning Resources

Coursera: Hands On Clinical Reporting Using R
Pharmaverse Examples: https://pharmaverse.github.io/examples/

This section has three questions: questions 1, 2 and 3.

Question 1: SDTM DS Domain Creation using {sdtm.oak}

{sdtm.oak} Learning Resources

https://pharmaverse.github.io/examples/ (SDTM section)
Slides and Training Videos https://pharmaverse.github.io/rinpharma-SDTM-workshop/
Package Documentation - https://pharmaverse.github.io/sdtm.oak/
CDISC SDTM Implementation Guide - Refer to the DS domain in the SDTMIG v3.4 in the CDISC Website. Refer to the PDF file named SDTMIG v3.4 in Files and Links tab.

Objective

Create an SDTM Disposition (DS) domain dataset from raw clinical trial data using the {sdtm.oak}.

Task

Develop an R program to create the DS domain using the below

Input raw data: `pharmaverseraw::ds_raw`

Study controlled terminology: The `study_ct` file is required to solve this exercise and you can get it by below options:

1) Download it from Github. or
2) If the Github link is not accessible, you can follow the instructions in Pharmaverse Running the example page and any of the examples in the SDTM section can provide the study_ct object. or
3) If 1) or 2) above doesn't work, create the required file using the instructions following the below code

```
None
study_ct <-
data.frame(
     stringsAsFactors = FALSE,
```

```
        codelist_code = c("C66727","C66727",
                          "C66727","C66727","C66727","C66727","C66727","C66727",
                          "C66727","C66727"),
            term_code = c("C41331","C25250",
                          "C28554","C48226","C48227","C48250","C142185","C49628",
                          "C49632","C49634"),
           term_value = c("ADVERSE EVENT",
                          "COMPLETED","DEATH","LACK OF EFFICACY","LOST TO FOLLOW-UP",
                          "PHYSICIAN DECISION","PROTOCOL VIOLATION",
                          "SCREEN FAILURE","STUDY TERMINATED BY SPONSOR",
                          "WITHDRAWAL BY SUBJECT"),
       collected_value = c("Adverse Event",
                          "Complete","Dead","Lack of Efficacy","Lost To Follow-Up",
                          "Physician Decision","Protocol Violation",
                          "Trial Screen Failure","Study Terminated By Sponsor",
                          "Withdrawal by Subject"),
  term_preferred_term = c("AE","Completed","Died",
                         NA,NA,NA,"Violation",
                          "Failure to Meet Inclusion/Exclusion Criteria",NA,"Dropout"),
         term_synonyms = c("ADVERSE EVENT",
                          "COMPLETE","Death",NA,NA,NA,NA,NA,NA,
                          "Discontinued Participation")
)
```

SDTM Programming details: This is the mock up eCRF that represents the data in `pharmaverseraw::ds_raw` and also has the programming details. Please refer to the`General Notes` in this PDF file.

Expected Result

An error-free program with good documentation that will create the DS domain with the following variables: STUDYID, DOMAIN, USUBJID, DSSEQ, DSTERM, DSDECOD, DSCAT, VISITNUM, VISIT, DSDTC, DSSTDTC, DSSTDY

Hint

This example is very similar to the AE example in the Pharmaverse Examples.

Deliverables

- SDTM creation script: `question_1_sdtm/01_create_ds_domain.R`
- Resulting SDTM dataset in any format
- A text file/log file as evidence for code running error-free

Question 2: ADaM ADSL Dataset Creation

{admiral} Learning Resources

Pharmaverse Examples - https://pharmaverse.github.io/examples/ (ADaM section)
Package Documentation - https://pharmaverse.github.io/admiral/

Objective

Create an ADSL (Subject Level) dataset using SDTM source data, the {admiral} family of packages, and tidyverse tools.

Task

Develop an R program to create the ADSL using the input SDTM data, the {admiral} family of packages, and tidyverse tools as explained in the Pharmaverse examples - ADSL or in {admiral} documentation. Adjust the logic and derive additional variables as mentioned below.

The DM domain is used as the basis of the ADSL. Start by assigning `pharmaversesdtm::dm` to an `adsl` object as explained in this section of the ADSL article.

Input datasets: `pharmaversesdtm::dm, pharmaversesdtm::vs, pharmaversesdtm::ex, pharmaversesdtm::ds, pharmaversesdtm::ae`

Derive the variables mentioned below:

Variable	Details
AGEGR9 & AGEGR9N	Age grouping into the following categories: “<18”, “18 - 50”, “>50”
TRTSDTM & TRTSTMF	Treatment start date-time (using the first exposure record for each participant and imputing missing hours and minutes but not seconds)
ITTFL	“Y”/”N” flag identifying patients who have been randomized, that is, where ARM is populated in pharmaversesdtm::dm domain.
LSTAVLDT	Last known alive date using any vital signs visit date, any adverse event start date, any disposition record and any exposure record.

The below section provides detailed specification to derive the above mentioned variables:

Variable	Specifications
AGEGR9	Age grouping into the following categories: “<18”, “18 - 50”, “>50”
AGEGR9N	Numeric Age grouping of Analysis Age [DM.AGE]. Categories are "<18", "18 - 50", ">50". Numeric groupings are 1, 2, 3.
TRTSDTM/TRTSTMF	Set to datetime of patient's first exposure observation Start Date/Time of Treatment [EX.EXSTDTC] converted to numeric datetime when sorted in date/time order. Derivation only includes observations where the patient received a valid dose (see NOTE) and datepart of Start Date/Time of Treatment [EX.EXSTDTC] is complete. If time is missing, ie. not collected, then impute completely missing time with 00:00:00, partially missing time with 00 for missing hours, 00 for missing minutes, 00 for missing seconds. **If only seconds are missing then do not populate the imputation flag (TRTSTMF).** **NOTE:** A valid dose is defined as (Dose per Administration [EX.EXDOSE] greater than 0 or (Dose per Administration [EX.EXDOSE] equal to 0 and Name of Actual Treatment [EX.EXTRT] contains 'PLACEBO')).
ITTFL	Set to "Y" if [DM.ARM] not equal to missing Else set to "N"
LSTAVLDT	Set to the last date patient has documented clinical data to show him/her alive, converted to numeric date, using the following dates: (1) last complete date of vital assessment with a valid test result ([VS.VSSTRESN] and [VS.VSSTRESC] not both missing) and datepart of [VS.VSDTC] not missing. (2) last complete onset date of AEs (datepart of Start Date/Time of Adverse Event [AE.AESTDTC]). (3) last complete disposition date (datepart of Start Date/Time of Disposition Event [DS.DSSTDTC]). (4) last date of treatment administration where patient received a valid dose (datepart of Datetime of Last Exposure to Treatment [ADSL.TRTEDTM]). Set to max of (Vitals complete, AE onset complete, disposition complete, treatment complete).

Expected Result

An error-free program with good documentation that will create the ADSL dataset with all the requested variables in the question. These should be derived using {admiral} functions where possible.

Hint

This additional variable derivation is very similar to the ADSL example in the Pharmaverse Examples: ADSL or in {admiral} documentation - https://pharmaverse.github.io/admiral/cran-release/articles/adsl.html

Deliverables

- ADSL creation script: `question_2_adam/create_adsl.R`
- Resulting ADaM dataset in any format
- A text file/log file as evidence for code running error-free

---

Question 3: TLG - Adverse Events Reporting

TLG Learning Resources

ggplot2 Documentation: https://ggplot2.tidyverse.org/index.html
FDA TLG Catalogue: https://pharmaverse.github.io/cardinal/quarto/index-catalog.html
Pharmaverse Examples: https://pharmaverse.github.io/examples/ - TLG section

Objective

Create outputs for adverse events summary using the ADAE dataset and {gtsummary}. This tests your ability to create regulatory-compliant clinical reports.

Input datasets: `pharmaverseadam::adae` and `pharmaverseadam::adsl`

Tasks

1. **Summary Table using {gtsummary} - HINT - FDA Table 10**
   Create a summary table of treatment-emergent adverse events (TEAEs).
   - Treatment-emergent AE records will have TRTEMFL == "Y" in `pharmaverseadam::adae`
   - Rows: AETERM or AESOC
   - Columns: Treatment groups (ACTARM)
   - Cell values: Count (n) and percentage (%)
   - Include total column with all subjects
   - Sort by descending frequency

   Sample Output:

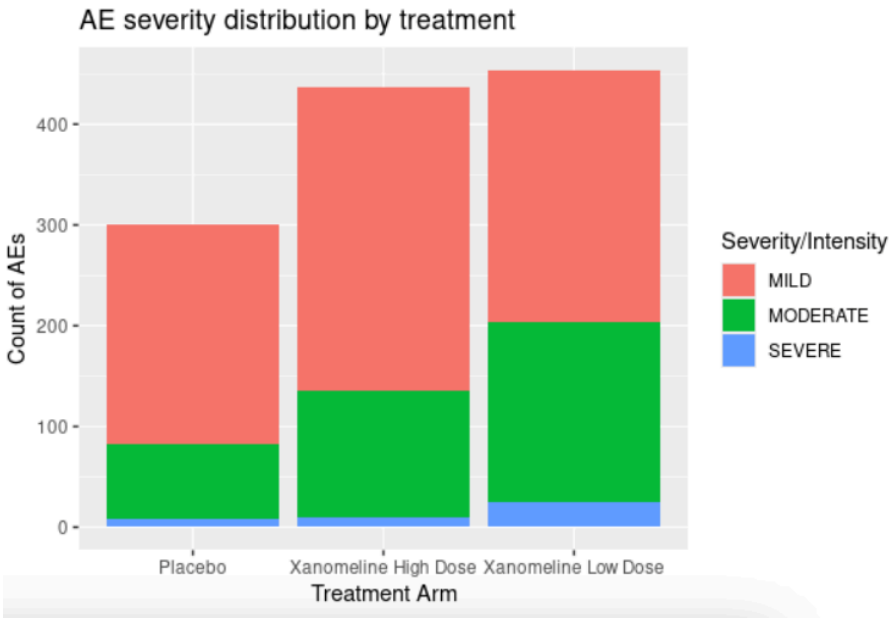
Primary System Organ Class Reported Term for the Adverse Event	Placebo N = 86[1]	Xanomeline High Dose N = 72[1]	Xanomeline Low Dose N = 96[1]
Treatment Emergent AEs	65 (76%)	68 (94%)	84 (88%)
CARDIAC DISORDERS	12 (14%)	14 (19%)	14 (15%)
ATRIAL FIBRILLATION	1 (1.2%)	2 (2.8%)	2 (2.1%)
ATRIAL FLUTTER	0 (0%)	1 (1.4%)	1 (1.0%)
ATRIAL HYPERTROPHY	1 (1.2%)	0 (0%)	0 (0%)
ATRIOVENTRICULAR BLOCK FIRST DEGREE	1 (1.2%)	0 (0%)	1 (1.0%)
ATRIOVENTRICULAR BLOCK SECOND DEGREE	1 (1.2%)	0 (0%)	0 (0%)

Primary System Organ Class Reported Term for the Adverse Event	Placebo N = 86[1]	Xanomeline High Dose N = 72[1]	Xanomeline Low Dose N = 96[1]
Treatment Emergent AEs	65 (76%)	68 (94%)	84 (88%)
GENERAL DISORDERS AND ADMINISTRATION SITE CONDITIONS	21 (24%)	36 (50%)	51 (53%)
APPLICATION SITE PRURITUS	6 (7.0%)	21 (29%)	23 (24%)
APPLICATION SITE ERYTHEMA	3 (3.5%)	14 (19%)	13 (14%)
APPLICATION SITE DERMATITIS	5 (5.8%)	7 (9.7%)	9 (9.4%)
APPLICATION SITE IRRITATION	3 (3.5%)	9 (13%)	9 (9.4%)
APPLICATION SITE VESICLES	1 (1.2%)	5 (6.9%)	5 (5.2%)
FATIGUE	1 (1.2%)	5 (6.9%)	5 (5.2%)

Output format: HTML/DOCX/PDF file

2. **Visualizations using {ggplot2)**
   - **Plot 1**: AE severity distribution by treatment (bar chart or heatmap). AE Severity is captured in the AESEV variable in `pharmaverseadam::adae` dataset.

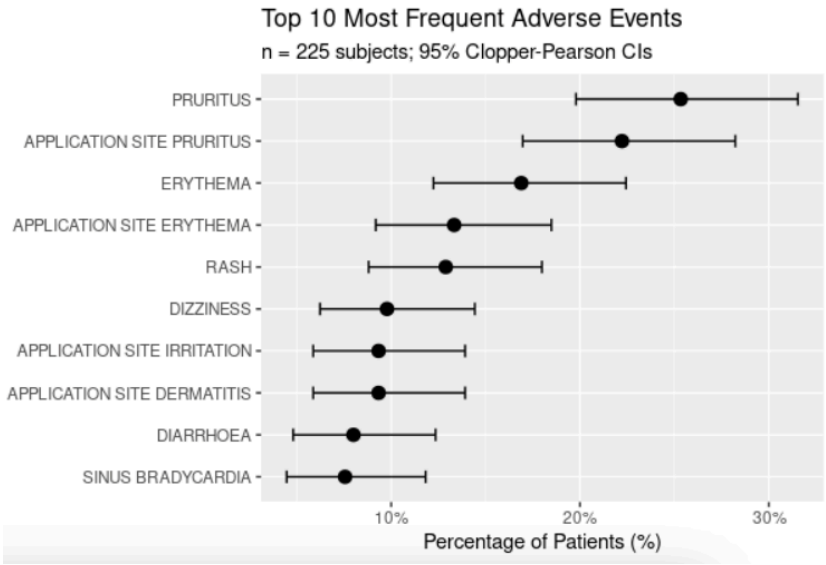
     Sample Output:



Output format: PNG file

- **Plot 2**: Top 10 most frequent AEs (with 95% CI for incidence rates). AEs are captured in the `AETERM` variable in the `pharmaverseadam::adae` dataset.

Sample Output:



Output format: PNG file

Deliverables

- Script to create summary table: `question_3_tlg/01_create_ae_summary_table.R`
- Script to create visualizations: `question_3_tlg/02_create_visualizations.R`
- Text files/log files as evidence for code running error-free

- Output files:
    - `ae_summary_table.html` (or .docx/.pdf)
    - Two PNG files (one for each output)

Python Coding Assessment

Question 4: GenAI Clinical Data Assistant (LLM & LangChain)

Objective

Develop a Generative AI Assistant that translates natural language questions into structured Pandas queries. The goal is to test your ability to use LLMs (e.g., OpenAI via LangChain) to dynamically map user intent to the correct dataset variable without hard-coding rules.

Scenario

A clinical safety reviewer wants to ask free-text questions about the AE dataset. They don't know the column names. Your Agent must "understand" the dataset schema and route the question to the correct variable. For example,

- If they ask about "severity" or "intensity" → Map to **`AESEV`**.
- If they ask about a specific condition (e.g., "Headache") → Map to **`AETERM`**.
- If they ask about a body system (e.g., "Cardiac", "Skin") → Map to **`AESOC`**.

Input

- **File:** `adae.csv (pharmaversesdtm::ae)`.
- **API Key:** You may use your own OpenAI API key or any other solution. If you do not have one, you may mock the LLM response in your code, but the logic flow (Prompt -> Parse -> Execute) must be complete.

Requirements

1. **Schema Definition:** Understand the data and define a dictionary or string in your code describing the relevant columns (`AESEV`, `AETERM`, `AESOC`, etc.) to the LLM.
2. **LLM Implementation:**
    - Create a function or class `ClinicalTrialDataAgent`.
    - Use an LLM to parse a user's question into a **Structured JSON Output** containing:
        - `target_column`: The column to filter.
        - `filter_value`: The value to search for (extracted from the question).
3. **Execution:**
    - Write a function that takes the LLM's output and applies the actual Pandas filter to the `ae` dataframe.

- Return the count of unique subjects (`USUBJID`) and a list of the matching IDs.

Deliverables

- **Code** of the solution developed
- **Test Script:** A simple block of code that runs 3 example queries (of your choice) and prints the results. An example question is, *`Give me the subjects who had Adverse events of Moderate severity.`*

FAQ & Troubleshooting

Q: Do I have to use the recommended packages like {sdtm.oak} for Question 1, 2 & 3?

A: Yes, it's recommended as it aligns with Pharmaverse best practices. Demonstrating familiarity with {sdtm.oak}, {admiral}, and {gtsummary} is a plus.

Q: What if I'm not familiar with CDISC standards?

A: Start by reviewing the CDISC resources listed above. The standards are well-documented, and the Pharmaverse examples provide excellent practical illustrations.