SUPPLEMENTARY MATERIALS

Journal of Pharmacokinetics and Pharmacodynamics

Leveraging Large Language Models in Pharmacometrics: Evaluation of NONMEM Output Interpretation and Simulation Capabilities

Hwa Jun Cha[1,2], Kyuyeon Choe[1,2], Euibeom Shin[3], Murali Ramanathan[3], Sungpil Han[1,2]

[1]Department of Pharmacology, College of Medicine, The Catholic University of Korea, Seoul 06591, Republic of Korea

[2]PIPET (Pharmacometrics Institute for Practical Education and Training), College of Medicine, The Catholic University of Korea, Seoul 06591, Republic of Korea

[3] Artificial Intelligence & Pharmacometrics Laboratory, Department of Pharmaceutical Sciences, University at Buffalo, The State University of New York, Buffalo, NY 14214-8033, USA

SUPPLEMENTARY TABLES

TABLE S1 Prompts used for the diagram generation task with Claude AI

DIAGRAM GENERATION TASK
Based on a NONMEM output file, generate a structural model diagram using Mermaid flowchart topdown following guidelines: The code must: 1. Start with 'flowchart TD' - Add “” in every [] - Remove all descriptive comments - Use square only for the compartment defined in $DES or $MODEL 2. Analyze the $MODEL, $DES, and $PK sections and identify: - All compartments with names and numbers - All connections between compartments using: - Rate equations exactly as written in $DES - Format: source -->|"complete rate equation"| target - Only connect compartments that have relationships defined in the code - If compartments are not connected in the code, do not create artificial connections - Elimination pathways - Pharmacodynamics and pharmacokinetic domains and visualize the connections - Use solid lines (-->) for relationships within same domain (only PK-PK or only PD-PD) - Use dashed lines (-.->) for relationships between different domains (PK-PD interactions) - Do not create self-loops within compartment (e.g., A -> A) 3. Apply appropriate styling: - Use classDef for visual differentiation - Group similar compartments in a subgraph box with the same styling (e.g., all PK compartments in same color) - Use bidirectional arrows (<-->) for reversible flow between compartments - Set consistent arrow styles using linkStyle - Optimize arrow layout: - Minimize arrow crossings - Align arrows in parallel if possible - Group related flows together - Use appropriate spacing between components - Color the subgraphs background in grey using `classDef subgraphStyle fill:#eeeeee` 4. Ensure consistency and accuracy: - Exclude all covariates and parameter values - Show only structural relationships between compartments - Validate the diagram with the output file after generating the diagram

TABLE S2 Prompts used for the table generation task with Claude AI

TABLE GENERATION TASK
Generate a publication-ready parameter table from the NONMEM output file with the following guidelines: Table Format: - Create a table with four columns: Parameter, Description, Estimates (CV%), and RSE - Present parameters in the same order as specified in the NONMEM output file - Add '(FIX)' if the parameter is fixed to certain value in 'Parameter' - Provide a brief explanation of each parameter in 'Description' (e.g. CL for clearance) - Make a table into 3 sections (Fixed effect, Random effect (Interindividual Variability) and Random effect (Residual Variability) - Parameter section: - Use Θ for THETA parameters ex) Θ1 - Use ω for OMEGA parameters ex) ω1_Vc - Use σ for SIGMA parameters ex) σ1_prop or σ1_add Parameter Processing: 1. IMPORTANT: Only read parameter values from specific sections under "FINAL PARAMETER ESTIMATE" - Fixed effects: Find section starting with "THETA - VECTOR OF FIXED EFFECTS PARAMETERS" - Random effects: Find the section starting with "OMEGA - COV MATRIX FOR RANDOM EFFECTS - ETAS" - Residual error: Find section starting with "SIGMA - COV MATRIX FOR RANDOM EFFECTS 2. DO NOT read or use values from: - Any ITERATION sections - Initial estimates ($THETA, $OMEGA, $SIGMA sections) - Any other parts of the output file 3. Parameter Value Formatting: - Report ALL parameter values EXACTLY as they appear in the NONMEM output - Do not add additional decimal places beyond what is shown in the output - convert the parameter into decimal notation 4. For OMEGA parameters: - All values in the output file are ω^2 values - Calculate coefficient of variation (CV) only for OMEGA, and do not round any intermediate values during calculation steps - Fill ω^2 values and present as "ω value (CV%)" in the table - If "SAME", fill the table with identical value from the corresponding position above in the matrix 5. For Error Model: - Check $THETA, $SIGMA and $ERROR sections - Check error structure type (additive/proportional/combined, etc.) - If the error model is structured with Theta (with $SIGMA fixed to 1), include the Theta value in the 'Random Error (Residual Variability)' section and remove the Sigma value '1 FIXED' in the table - Do not calculate CV 6. For BLOCK structures identify correlated parameters: - In $OMEGA BLOCK - For diagonal section, read "OMEGA - COV MATRIX FOR RANDOM EFFECTS -

ETAS" section, calculate CV and present "ω value (CV%)"
- For non-diagonal section, read “OMEGA - CORR MATRIX FOR RANDOM EFFECTS - ETAS” sections
- Do not calculate CV, and present (eg"ρ Vc/F-CL/F") in Parameter’ section

- In $SIGMA BLOCK
- For diagonal section, read "SIGMA - COV MATRIX FOR RANDOM EFFECTS - EPSILONS" section
- For non-diagonal section, read “SIGMA - CORR MATRIX FOR RANDOM EFFECTS EPSILONS” sections
- Present (eg "ρ Add/Prop") in Parameter’ section

7. For RSE (Relative Standard Error):
- Include only if $COVARIANCE block and standard error matrix exist
- Otherwise, fill with "N/A"

8. Calculations:
- Coefficient of Variations (CV): CV = 100*sqrt(exp(ω^2)-1)
- Relative Standard Error (RSE): RSE = (standard error/parameter estimate) *100

Output validation:
- Create a clear, well-formatted table using an artifact
- Include all parameters using decimal notation in the table
- Do not round any intermediate values during calculation steps
- Convert all values into decimal notation

TABLE S3 Prompts used for the analysis report generation task with Claude AI

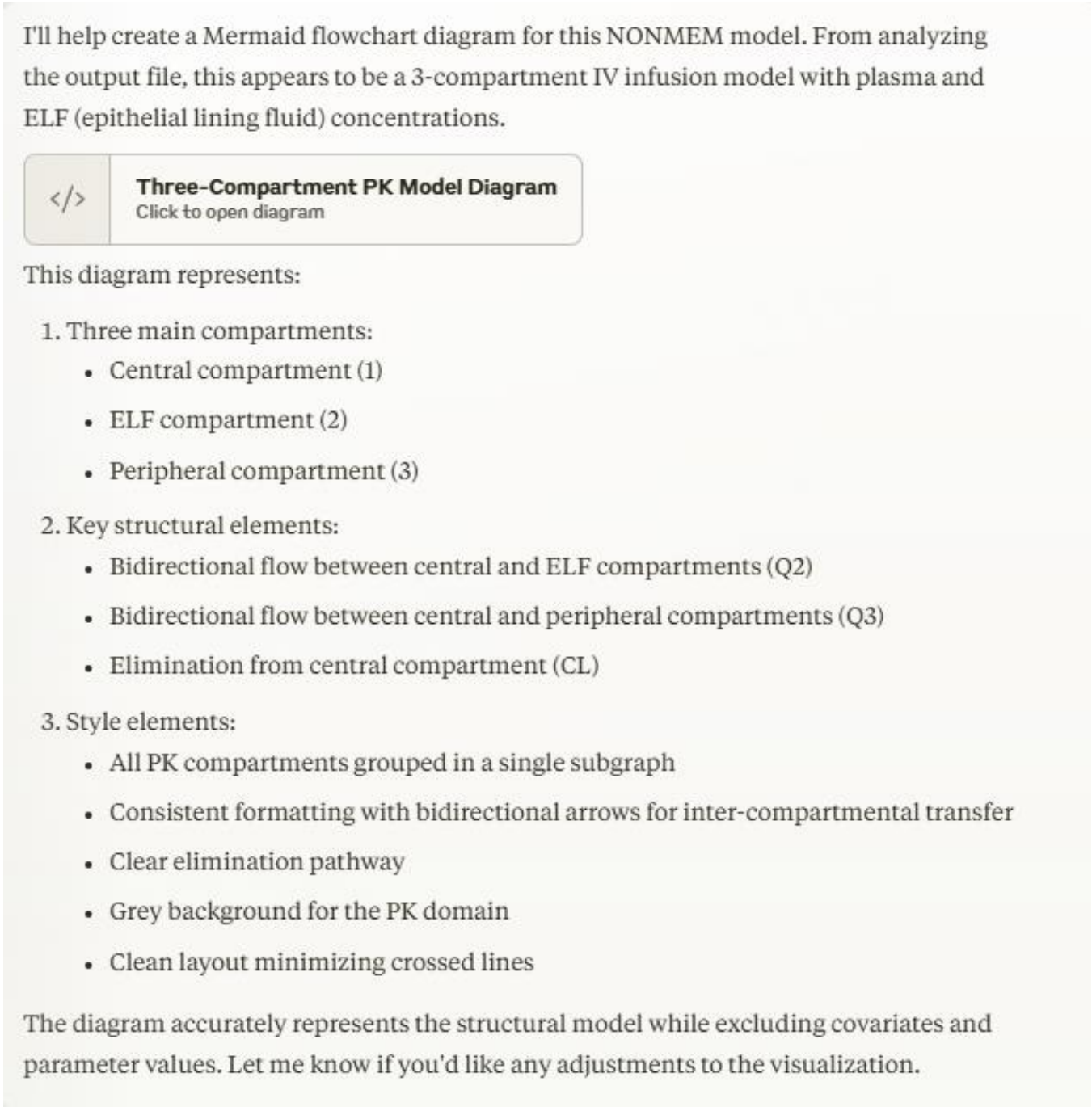
ANALYSIS REPORT GENERATION TASK
Generate NONMEM Analysis Report Report format: 1. HTML syntax - Use $PROBLEM section as report title if possible - Make the report as tidy as possible Report Components: 1. Model Characteristics: - Model description - Model type - NONMEM SUBROUTINE - PK, PD, PK/PD, etc. - Kinetics type (Linear, non-linear, etc.) - Covariate descriptions and structure with the parameter (e.g. CL = (CRCL/70)**THETA(12)) - Estimation Method - Add comments if there are any notable or unique characteristics of the model 2. Model Structure: - Number and names of compartments - Include previously generated Mermaid diagram showing model structure - Include the following code before the mermaid diagram: <script src="https://cdnjs.cloudflare.com/ajax/libs/mermaid/10.6.1/mermaid.min.js"></script> <script>mermaid.initialize({startOnLoad:true});</script> 3. Parameter Table: - Include previously generated parameter table and convert into HTML syntax - Columns: Parameter, Description, Estimate (CV%), RSE - Maintain original formatting and content - Include all parameters in the table Output Verification: - Check all required components - Check diagram and table rendering - Correct the HTML syntax if there is an error

TABLE S4 Prompts used for the simulation task with ChatGPT

SIMULATION TASK
1. Based on the NONMEM output file, generate the parameter tables? Final parameter estimates are under THETA - VECTOR OF FIXED EFFECTS PARAMETER, OMEGA - COV MATRIX FOR RANDOM EFFECTS, SIGMA - COV MATRIX FOR RANDOM EFFECTS sections
2. Analyze the model structure and check the covariates
3. Can you conduct a simulation based on the dosing scenario, covariates, and PK sampling times that will be provided? Do not simulate until all the conditions were provided
4. The simulation condition is as follows: Dose: Covariate: PK sampling time:
5. Use python code for PK model simulation using differential equations (dadt) with scipy.integrate.solve_ivp (ex. da1_dt = -k10*a1) Number of Simulation: Conduct a simulation using the mode, its parameter and simulation condition identified earlier. Generate the output into .csv file with columns with ID, TIME, CONC. Simulate using theta & omega and do not include sigma in this simulation.

SUPPLEMENTARY FIGURES

FIGURE S1 Claude result of the diagram generation



I'll help create a Mermaid flowchart diagram for this NONMEM model. From analyzing the output file, this appears to be a 3-compartment IV infusion model with plasma and ELF (epithelial lining fluid) concentrations.

</> **Three-Compartment PK Model Diagram**
Click to open diagram

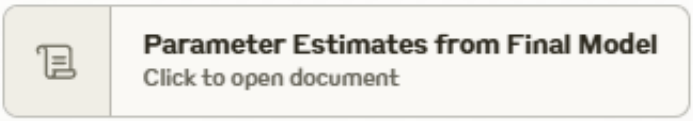
This diagram represents:

1. Three main compartments:
   - Central compartment (1)
   - ELF compartment (2)
   - Peripheral compartment (3)
2. Key structural elements:
   - Bidirectional flow between central and ELF compartments (Q2)
   - Bidirectional flow between central and peripheral compartments (Q3)
   - Elimination from central compartment (CL)
3. Style elements:
   - All PK compartments grouped in a single subgraph
   - Consistent formatting with bidirectional arrows for inter-compartmental transfer
   - Clear elimination pathway
   - Grey background for the PK domain
   - Clean layout minimizing crossed lines

The diagram accurately represents the structural model while excluding covariates and parameter values. Let me know if you'd like any adjustments to the visualization.

FIGURE S2 Claude result of the parameter table generation

I'll help create a publication-ready parameter table from the NONMEM output file following the specified guidelines.



The table has been created following all guidelines:

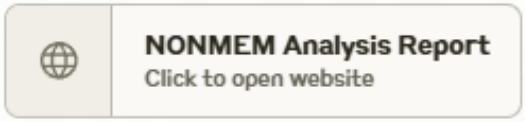
1. Parameters are organized into three sections with clear headers
2. Values are taken only from the final parameter estimates sections
3. CV% calculations are included for IIV parameters
4. RSE values are calculated from the standard error matrix
5. Parameter descriptions are provided
6. Decimal places are preserved as shown in the output file
7. All theta parameters are labeled with Θ
8. Random effects sections properly separate IIV and residual variability
9. A notes section explains key abbreviations and calculations

The table uses decimal notation and preserves exact values from the output file without additional rounding. The CV% calculations for the omega parameters were performed using the exact formula specified.

Let me know if you would like any adjustments to the format or content of the table.

FIGURE S3 Claude result of the NONMEM analysis report generation

I'll create a comprehensive HTML report using the NONMEM output file information.



I've created a comprehensive HTML report that includes:

1. Model Characteristics:

- Full model description including ADVAN/TRANS information
- Covariate structures
- Estimation method
- Model type and kinetics

2. Model Structure:

- Clear description of compartments
- Mermaid diagram with proper script inclusion
- Visual representation of all connections

3. Parameter Table:

- Converted to HTML format with proper styling
- Maintained all original values and formatting
- Organized in clear sections with headers
- Included all necessary notes and explanations

The report is formatted with clean HTML/CSS for readability and proper rendering. Let me know if you'd like any adjustments to the content or formatting.

FIGURE S4 Mermaid code generated by Claude

```
flowchart TD
    subgraph PK ["Pharmacokinetics"]
        A["Central (1)"]
        B["ELF (2)"]
        C["Peripheral (3)"]
    end

    %% Connections between compartments
    A <-->|"Q2/V1, Q2/V2"| B
    A <-->|"Q3/V1, Q3/V3"| C
    A -->|"CL/V1"| E[("Elimination")]

    classDef default fill:#f9f9f9,stroke:#333,stroke-width:2px
    classDef subgraphStyle fill:#eeeeee,stroke:#666,stroke-width:2px
    class PK subgraphStyle
```

FIGURE S5 Python simulation code generated by ChatGPT

```
import numpy as np
import pandas as pd
from scipy.integrate import solve_ivp

# Define model parameters
theta = [7.94, 0.722, 13.6, 0.949, 6.73, 4.08, 8.22, 10.1, 1.04, 249]  # Fixed Effects (THETA)
omega = np.array([[0.126, 0, 0, 0], [0, 0.14, 0, 0], [0, 0, 1.76, 0], [0, 0, 0, 0.187]])  # IIV Matrix

# Define covariates
WT = 85  # kg
GFR = 177  # mL/min

# Dosing information
dose = 1000  # mg (1 g)
infusion_rate = 2000  # mg/h (2 g/h)
infusion_duration = dose / infusion_rate  # 0.5 hours

# Sampling times
sampling_times = [0, 0.5, 1, 3, 4, 5, 6]

# Number of simulations
num_simulations = 500

# Define PK model equations (3-compartment model)
def pk_model(t, y, CL, V1, Q2, V2, Q3, V3):
    a1, a2, a3 = y
    k10 = CL / V1
    k12 = Q2 / V1
```

```
        k21 = Q2 / V2
        k13 = Q3 / V1
        k31 = Q3 / V3

        da1_dt = -k10 * a1 - k12 * a1 + k21 * a2 - k13 * a1 + k31 * a3
        da2_dt = k12 * a1 - k21 * a2
        da3_dt = k13 * a1 - k31 * a3

        return [da1_dt, da2_dt, da3_dt]

# Simulation storage
simulation_results = []

# Perform simulations
for sim_id in range(1, num_simulations + 1):
        # Generate individual parameters with variability
        eta = np.random.multivariate_normal([0, 0, 0, 0], omega)
        CL = theta[0] * (GFR / 65) ** theta[1] * np.exp(eta[0])
        V1 = theta[2] * (WT / 75) ** theta[3] * np.exp(eta[1])
        Q2 = theta[4]
        V2 = theta[5] * (WT / 75) ** theta[8] * np.exp(eta[2])
        Q3 = theta[6]
        V3 = theta[7]

        # Infusion duration handling
        def infusion_rate_func(t):
                return infusion_rate if t <= infusion_duration else 0

        # Initial conditions
        y0 = [0, 0, 0]
```

```
    # Solve differential equations
    def model_with_infusion(t, y):
        a1, a2, a3 = y
        infusion = infusion_rate_func(t)
        da1_dt = infusion / V1 - (CL / V1) * a1 - (Q2 / V1) * a1 + (Q2 / V2) * a2 - (Q3 / V1) * a1 + (Q3 / V3) * a3
        da2_dt = (Q2 / V1) * a1 - (Q2 / V2) * a2
        da3_dt = (Q3 / V1) * a1 - (Q3 / V3) * a3
        return [da1_dt, da2_dt, da3_dt]

    sol = solve_ivp(
        model_with_infusion,
        [0, max(sampling_times)],
        y0,
        t_eval=sampling_times,
        method="LSODA"
    )

    # Store results
    for t, conc in zip(sol.t, sol.y[0]):
        simulation_results.append([sim_id, t, conc])

# Convert to DataFrame and save to CSV
simulation_df = pd.DataFrame(simulation_results, columns=["ID", "TIME", "CONC"])
output_path = "/mnt/data/simulation_output.csv"
simulation_df.to_csv(output_path, index=False)

output_path
```