МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет імені В.Н.Каразіна
Факультет математики і інформатики
Кафедра теоретичної та прикладної інформатики

Кваліфікаційна робота

магістр

на тему «Використання SAS Macro для автоматизації роботи у фармацевтичній галузі»

Виконав: студент 2 курсу, групи МФ-61
спеціальність 122 «Комп'ютерні науки»
освітньо-наукова програма «Інформатика»

Лозінська О.Я.
(прізвище та ініціали)

Керівник _____
(прізвище та ініціали)

Рецензент Левчук І. П.
(прізвище та ініціали)

Харків – 2024

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ADaM – Analysis Data Model

ADSL – Subject-Level Analysis Dataset

AE – Adverse Events

ATC – Anatomical Therapeutic Chemical

BDS – Basic Data Structure

CDASH – Clinical Data Acquisition Standards Harmonization

CDISC – Clinical Data Interchange Standards Consortium

CRF – Case Report Form

CM – Concomitant Medications

CRO – Contract Research Organization

CSV – Comma-Separated Values

CT – Controlled Terminology

DM – Demographics

eCRF – Electronic Case Report Form

EMA – European Medicines Agency

EX – Exposure

FDA – U.S. Food and Drug Administration

GCP – Good Clinical Practice

HTML – Hyper Text Markup Language

ICH-GCP – International Council for Harmonisation of Technical Requirements for Pharmaceuticals for Human Use – Good Clinical Practice

IG – Implementation Guide

LB – Laboratory Test Results

MedDRA – Medical Dictionary for Regulatory Activities

MH – Medical History

OCCDS – Occurrence Data Structure

ODM – Operational Data Model

PDF – Portable Document Format

PDV – Program Data Vector

PMDA – Pharmaceuticals and Medical Devices Agency

PR – Procedures

PRM – Protocol Representation Model

PT – Preferred Term

QC – Quality Control

SAP – Statistical Analysis Plan

SAS – Statistical Analysis System

SDTM – Study Data Tabulation Model

SOC – System Organ Class

TLF – Tables, Listings, Figures

VS – Vital Signs

WHODD – World Health Organization Drug Dictionary

XML – Extensible Markup Language

ВСТУП

1.1 Область дослідження та застосування

Автоматизація процесів є актуальною проблемою сьогодення у різних сферах діяльності, зокрема, у такій важливій галузі, як клінічні дослідження (випробовування). Клінічні дослідження є невід'ємною складовою медичного наукового світу, так як забезпечують ефективність та безпеку застосування нововпроваджених препаратів.

Клінічні дослідження – це наукові дослідження (детальніше див. розд. 1.2 та 1.3), які проводяться з метою вивчення дії препаратів, медичних процедур, стратегій лікування тощо (далі препаратів) на людях [1]. Усі дослідження повинні проводитись з дотриманням міжнародних вимог та стандартів, таких як, GCP (Good Clinical Practice), ICH-GCP (International Council for Harmonisation of Technical Requirements for Pharmaceuticals for Human Use – Good Clinical Practice). Дотримання цих стандартів забезпечує високий рівень довіри до результатів клінічних випробувань, що є важливим для прийняття рішень щодо безпеки та ефективності лікарських засобів, а також для визнання досліджень в міжнародному масштабі [2].

Обробка результатів клінічних досліджень відбувається в декілька етапів: збір сирих (необроблених) даних, обробка сирих даних та їхнє представлення у стандартизованих SDTM-датасетах (Study Data Tabulation Model), приготування стандартизованих даних для потреб аналізу – ADaM-датасетів (Analysis Data Model), та виведення фінальних результатів виконаного аналізу у вигляді звітів TLF (Tables, Listings, Figures). Датасетами, відповідно до CDISC-термінології [3], називають набори структурованих даних в одному файлі.

TLF-звіти є важливою частиною документації клінічних випробувань, яка використовується для представлення результатів до регуляторних органів (FDA, EMA, PMDA тощо). TLF – це таблиці, лістинги та графіки, які використовуються

в клінічних дослідженнях для аналізу та візуалізації даних та допомагають дослідникам та регуляторним органам зрозуміти результати дослідження, а саме, оцінити ефективність та безпечність нового препарату, а також порівняти результати з попередніми дослідженнями.

TLF-звіти по побічним ефектам (AE – Adverse Events) є стандартною та обов'язковою частиною кожного клінічного випробування і мають типову структуру. Вони містять інформацію про небезпечні події (AE), що виявлені під час дослідження медичних препаратів, таку як тип події, ступінь тяжкості, потенційний зв'язок з досліджуваним препаратом та інші важливі деталі. В даній роботі розглянуто такий тип таблиць в яких виконується розрахунок частот подій та їхніх відсотків для кожної досліджуваної стратифікаційної групи.

Автоматизація процесів клінічних досліджень має кілька важливих задач. В першу чергу це ефективність та швидкість роботи клінічних програмістів, тобто спеціалістів, що напряму займаються обробкою даних клінічних досліджень та створенням TLF-звітів. Програмісти часто стикаються з запитами, які або повторюються, або потребують генерування вихідних даних за схожими шаблонами. Створення окремих програм для кожного результату займає багато часу, особливо коли потрібні часті оновлення та відправки цих результатів регуляторним органам. Отже, вирішення цієї проблеми полягає в прийнятті надійної та широко використовуваної програми або інструменту, який полегшить робоче навантаження. Друга задача полягає в зменшенні впливу людського фактору при обробці результатів, так як при великій кількості оновлень даних та програм виникає багато помилок. Обсяги даних можуть бути доволі масивними, тому автоматизація дозволяє більш точно відстежувати зміни та знаходити неточності.

В дипломній роботі представлено макроінструмент SAS (SAS-макрос), призначений для ефективного розв'язання цих повторюваних запитів та для спрощення генерації таблиць та датасетів, що є важливим завданням при аналізі даних клінічних досліджень. Цей інструмент не лише спрощує та пришвидшує процес, але також має потенціал для подальшого вдосконалення, щоб

пристосовуватися до різних ситуацій, дозволяючи користувачеві створювати таблиці-звіти, що місять описові дані, підрахунки та відсотки. Інструмент є зручним у використанні, вимагаючи від користувачів лише заповнити необхідні параметри і викликати макрос. Його універсальність також полягає в тому, що він може працювати як з SDTM-, так і з ADaM-датасетами.

1.2 Клінічні дослідження

Клінічні випробування (дослідження) − це наукові дослідження, основною метою яких є вивчення властивостей препарату, обладнання або запропонованого методу лікування [1]. Суб’єктами таких досліджень є люди. Виявлення впливу досліджуваної речовини на організм пацієнтів та оцінка ефективності її дії проти клінічних проявів захворювання являють собою першочергове завдання будь-якого клінічного випробування.

Клінічне дослідження є обов'язковим етапом при розробці нових методів лікування. Правомірність його проведення контролюється такими міжнародними організаціями як FDA (United States Food and Drug Administration), PMDA (Japanese Pharmaceuticals and Medical Devices Agency), EMA (European Medicines Agency), а достовірність позитивних результатів виступає основним фактором для надання препарату права виходу на ринок [3].

1.3 Процес проведення клінічних досліджень

Тестування досліджуваного препарату розпочинається з набору групи добровольців певної кількості. Дизайн клінічного дослідження має на меті забезпечити наукову достовірність отриманих результатів, а тому початкові фази випробування, розраховані на визначення безпечності препарату, визначення ефективної дози та ідентифікацію побічних ефектів, включають малочисельні групи пацієнтів, однак згодом проводять все більш масштабні порівняльні дослідження для оцінки ефективності.

Клінічні випробування можуть включати в себе як здорових людей, так і людей з певними захворюваннями, які бажають спробувати нові методи лікування. Варто зазначити, що здорові добровольці набираються тільки на першій фазі випробування. Для того щоб взяти участь у дослідженні, суб'єкт має відповідати затвердженому дослідниками переліку характеристик, які називаються критеріями включення. Це може бути наявність певної стадії захворювання, сімейний анамнез, приналежність до визначеної вікової групи або статі, історія зловживання забороненими речовинами тощо. Також разом із критеріями включення визначається набір критеріїв виключення – учасникам, що володіють певними характеристиками, буде заборонено брати участь у клінічному випробуванні. Критерії виключення являють собою такі фактори, що можуть перешкоджати ефективній дії досліджуваного препарату або становити загрозу для життя і здоров'я суб'єктів на випробуванні.

Після того, як пацієнти набрані, їх розділяють на групи методом випадкового відбору (рандомізують). Ці групи поділяються на: групу, в якій суб'єкти приймають ліки з досліджуваним препаратом, групу з порівняльним препаратом або плацебо. Зазвичай золотим стандартом є рандомізоване подвійне сліпе плацебо-контрольоване клінічне дослідження, де «подвійне сліпе» означає, що ані лікар, ані пацієнт не знають, що приймає пацієнт протягом випробування.

Впродовж клінічного дослідження суб'єкт під контролем фахівців приймає діючий препарат або плацебо відповідно до групи в яку він рандомізований. Під час візитів до медичних центрів, проводяться потрібні аналізи та опитування, результати яких фіксуються і надалі обробляються клінічними програмістами.

Після того, як всі етапи клінічного дослідження завершені, дослідники мають можливість проаналізувати отримані дані та зрозуміти чи є ефективним та безпечним препарат, що тестувався. Якщо результат позитивний, препарат може перейти на вищу фазу дослідження та в кінцевому результаті вийти на ринок.

Впродовж дослідження фахівці збирають велику кількість клінічно важливої інформації за допомогою Case Report Form (CRF). Демографічні дані пацієнтів, результати лабораторних аналізів, свідчення про побічні ефекти, інформація про прийом супутніх ліків, які відбулися за час випробування, різноманітні опитування, тощо – усе це складає основу статистичного аналізу клінічного дослідження, результати якого, зрештою, і визначать ефективність та безпечність запропонованого препарату. Але для того, щоб математично проаналізувати зібрані дані, їх попередньо потрібно підготувати, а саме – обробити із дотриманням міжнародних загальноприйнятих стандартів.

1.4 Використовувані стандарти

Для обробки зібраних (сирих) даних використовуються стандарти, які були розроблені CDISC (Clinical Data Interchange Standards Consortium). CDISC – це міжнародна організація, яка займається розробкою та регуляцією стандартів в галузі клінічних досліджень. CDISC та FDA тісно співпрацюють з моменту заснування CDISC, щоб забезпечити стандарти даних, які дозволяють регуляторним експертам більш ефективно отримувати, обробляти, перевіряти та архівувати подані документи [4]. Дотримання стандартів CDISC є обов'язковими для регуляторних подань до FDA.

В основному, стандарти CDISC (рис. 1) поділяються на чотири групи: PRM (Protocol Representation Model), CDASH (Clinical Data Acquisition Standards Harmonization), SDTM (Study Data Tabulation Model), ADaM (Analytical Data Model). Це базові стандарти для багатьох інших стандартів CDISC, які підтримують збір, управління, аналіз і звітність даних на всіх етапах процесу клінічного дослідження.

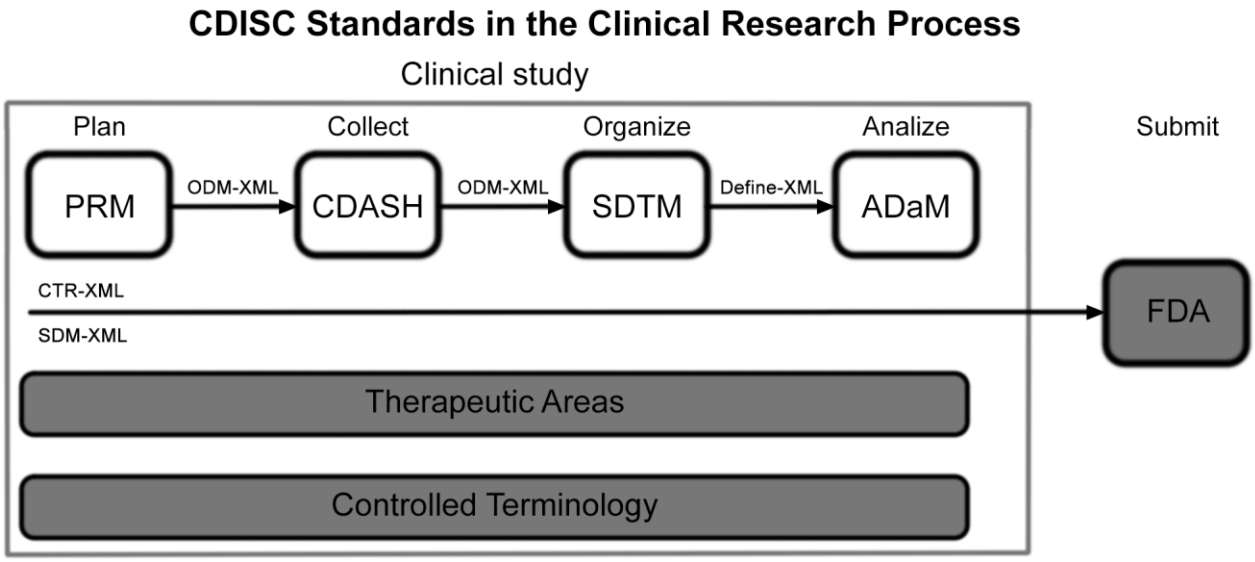

Рисунок 1 – Стандарти CDISC використовувані в процесі клінічних досліджень [5]

Клінічне дослідження розпочинається з розробки дизайну та плану дослідження. Головним документом дослідження є протокол, в якому визначені дизайн, ціль та організація дослідження.

Для роботи з протоколом використовується модель PRM [6]. Ця модель дає можливість фіксувати інформацію з протоколу у вигляді даних, також за допомогою PRM створюється CRF. CRF – це документ, призначений для запису всієї інформації про пацієнта, яка необхідна під час клінічного випробування. CRF може бути як в електронній так і в паперовій формі. При цьому використовуються стандарти CDASH [7].

CDASH описує базову структуру для організації CRF, іменування, опис змінних і пов'язаних з ними атрибутів для підтримки збору даних у клінічних випробуваннях [8].

Після того, як клінічні дані зібрано, їх необхідно організувати в таблиці (датасети, набори даних). Для цього використовується SDTM-стандарт, за допомогою якого дані записуються в SDTM-датасет, де кожен рядок є окремим фрагментом даних, а стовпці є змінними. SDTM-стандарт описує всі змінні, які дозволено використовувати в при створені SDTM-датасетів, а також їхні метадані – назву, тип, довжину та формат.

Therapeutic areas – стандарт для різних медичних галузей, наприклад, він включає в себе посібник, який пояснює, як збирати дані та проводити випробування, пов'язані з COVID-19.

Далі, коли SDTM-датасети готові, їх використовують для створення датасетів статистичного аналізу. Для цього використовується ADaM-стандарт.

ADaM-стандарт описує основні принципи для створення статистичних ADaM-датасетів. Набори даних ADaM повинні бути готовими до аналізу і містити простежуваність між ADaM та SDTM-датасетами [9].

На етапі створення SDTM- та ADaM-датасетів дані кодуються згідно до контрольної термінології (CT, Controlled Terminology). Контрольна термінологія – це структурований набір даних, що містить перелік термінів, які слід використовувати зі стандартами CDISC [10]. Таким чином, «всі» хто знайомі зі стандартом можуть легко зрозуміти записані до датасетів дані. ADaM-датасети мають більш вільну форму представлення даних, використання контрольної термінології у більшості випадків не є обов’язковим. Це, в першу чергу, пов’язане з головною вимогою до ADaM-датасетів, а саме підтримка аналізу.

Останнім етапом обробки клінічних даних є створення TLF-звітів. TLF-звіти – це набори даних клінічного дослідження, які допомагають проаналізувати та узагальнити набори даних клінічного дослідження виведені в зручному стислому форматі. Створення TLF-звітів виконуються на основі Statistical Analysis Plan (SAP). SAP – документ, в якому більш детально прописані заплановані статистичні методи та інструкції для програмістів по обробці даних.

Коли весь набір SDTM, ADaM і TLF виконаний та провалідований, його можна відправляти до регуляторного органа, наприклад FDA. Для обміну даних та метаданих між різними етапами в клінічних дослідженнях використовується ODM (Operational Data Model). ODM використовує XML (Extensible Markup Language) для обміну даними в усій системі охорони здоров'я. XML дозволяє легко трансформувати його в PDF, Word, HTML та інші формати.

Для перевірки даних на відповідність CDISC-стандартам використовується OpenCDISC Validator Pinnacle 21. Pinnacle 21 виконує перевірку наборів даних

SDTM та ADaM, а також Define XML та видає звіт у форматі Excel або CSV, який містить інформацію про помилки, які необхідно виправити перед подачею до регулятивних органів[11].

1.5 Процес обробки даних клінічними програмістами

Процес обробки даних розпочинається зі збору даних за допомогою CRF. Формат CRF визначається дизайном клінічного дослідження та вимогам, що зазначені у протоколі цього клінічного дослідження. В CRF форму вносяться всі можливі дані, які збираються протягом клінічного випробування. Дані пацієнтів вносяться безпосередньо в лікарнях або лабораторіях, де пацієнт проходить обстеження, або до електронної форми (eCRF), наприклад, у виданому пацієнтові планшету. Ці дані включають в себе вік, вагу, дані лабораторних досліджень, побічні ефекти, медичну історію перенесених захворювань, препарати, які приймає чи приймав раніше суб’єкт та ін.

Після того, як CRF проанотовано командою клінічних програмістів розпочинається робота над створенням SDTM-датасетів. Процес анотації є обов'язковим та надає розуміння до якої з змінних SDTM-датасетів записується та чи інша інформація, що зібрана у CRF.

SDTM-датасети мають вертикальну структуру і повинні відповідати стандартам CDISC, які визначають назву кожної змінної, тип та формат. Документація SDTM не визначає довжину змінних, але важливо дотримуватися узгодженості довжини змінних з однаковими назвами в різних доменах (збірках спостережень, об'єднаних певною тематичною спільністю) дослідження. Створені набори даних будуть використані в подальшому при розробці ADaM-датасетів. Але в деяких випадках можуть також використовуватись напряму для розробки TLF.

Для створення SDTM-датасету програміст користується SDTM-стандартами, контрольною термінологією (CT) та специфікацією, яка створюється перед початком роботи над SDTM-датасетами. Специфікація на

датасет має опис кожної змінної цього датасету, її метаданих, використовуваної контрольної термінології та, що найголовніше, містить відомості щодо алгоритмів та методів отримання того чи іншого значення кожної з наявних змінних.

Кожен SDTM-домен призначений для зберігання однотипних даних таких як, наприклад, демографічні дані (DM, Demographics), показники життєдіяльності (VS, Vital Signs), побічні ефекти (AE, Adverse Events), лабораторні дані (LB, Laboratory Test Results), дані щодо дозування досліджуваним препаратом чи плацебо (EX, Exposure) тощо. Таким чином, сукупність SDTM-датасетів – є набором логічно пов'язаних між собою спостережень із загальною, конкретною темою, які зазвичай збираються для всіх суб'єктів клінічного дослідження. Логіка взаємозв'язку може стосуватися наукової тематики даних або їхньої ролі у дослідженні . На даний момент SDTM-стандарт налічує більше 30 доменів [12]. Однак слід зазначити, що в кожному конкретному клінічному дослідженні створюються тільки ті домени, які є необхідними конкретному саме для цього дослідження.

Назви, опис та класифікація деяких, найбільш використовуваних доменів наведено у табл. 1.

Кожен тип домену має свої змінні, які можуть бути включені в цей домен – не всі змінні, що включені до SDTM-стандарту повинні використовуватись, так як вони можуть бути не обов’язковими для конкретного дослідження.

Деякі обмеження, що накладаються на довжини змінних та їхніх метаданих, зумовлені використанням транспортного протоколу SAS верії 5 (є обов'язковим): ім’я змінної – 8, лейбл (короткий опис змінної) – 40, символьна змінна – 200. Крім того, деякі змінні можуть мати тільки ті значення, які присутні в контрольній термінології, а текст (вміст) змінних, які використовують контрольну термінологію, рекомендовано писати літерами верхнього регістру.

CDISC SDTM DOMAINS

CLASS	DOMAIN NAME	DOMAIN DESCRIPTION
Special-Purpose	CO	Comments
	SE	Subject Elements
	DM	Demographics
	SV	Subject Visits
Interventions	CM	Concomitant Medications
	EX	Exposure
	EC	Exposure as Collected
	PR	Procedures
Events	AE	Adverse Events
	CE	Clinical Events
	DS	Disposition
	MH	Medical History
Findings	EG	ECG Test Results
	PE	Physical Examination
	VS	Vital Signs
	DD	Death Details
	IE	Inclusion/Exclusion Criterion Not Met
	LB	Laboratory Test Results
	QS	Questionnaires
Trial Design	TA	Trial Arms
	TE	Trial Elements
	TI	Trial Inclusion/Exclusion Criteria
	TV	Trial Visits
	TS	Trial Summary
Relationship Datasets	SUPP-- datasets	Supplemental Qualifiers
	RELREC	Related Records

Таблиця 1 – Назви, опис та класифікація деяких, найбільш використовуваних SDTM-доменів

Кожен SDTM-датасет, після його створення, обов'язково сортується за унікальним ключем, який складається з переліку змінних, значення яких відповідають тільки одному єдиному запису в цьому датасеті. Для DM-датасету, наприклад, це змінна USUBJID – унікальний ідентифікатор суб'єкта клінічного дослідження.

У зв'язку з тим, що ADaM-датасети призначенні саме для використання при створенні TLF-звітів (фактично аналізу даних), їхнім головним завданням є підтримання ефективної генерації, реплікації та перегляду статистичних аналізів клінічних випробувань, а також впровадження простежуваності між

результатами аналізу, даними аналізу та даними, що представлені в моделі табуляції даних дослідження – SDTM-датасетах [13].

Процес створення ADaM-датасетів схожий за своєю концепцією до створення SDTM-датасетів. Програмісти створюють датасети та специфікацію до них, використовуючи ADaM-стандарти. ADaM-датасети містять в собі статистично оброблені SDTM-дані, які будуть використовуватись в TLF-звітах. Наразі ADaM-стандарт має 3 структури: ADSL, BDS і OCCDS, які відповідають класам SUBJECT LEVEL ANALYSIS DATASET, BASIC DATA STRUCTURE і OCCURRENCE DATA STRUCTURE наборів даних ADaM. Набори даних для аналізу, які відповідають основним принципам ADaM та іншим конвенціям ADaM, але не відповідають одній з 3 визначених структур (ADSL, BDS, OCCDS), вважаються наборами даних ADaM з класом ADAM OTHER. Приклад структури ADaM датасетів наведено на Рис. 2 [13].

Особливою відмінністю між SDTM- та ADaM-датасетами є форма представлення дат. В SDTM-датасетах змінні, що містять дати та час є символьними, а значення в них відформатоване відповідно до стандарту ISO-8601. Наприклад, якась подія, яку зафіксовано о сьомій годині ранку 13 го липня 1983 року, буде супроводжуватися SDTM-змінною дати-часу, значення якої матиме вигляд 1983-07-13T07:00. В ADaM-датасетах, дати та час завжди зберігаються у числових змінних, значенням яких є ціле число. На це число накладається формат відображення SAS (назва формату – DATE9), таким чином число 1 інтерпретується та відображається (за замовчуванням) як 1 січня 1960 року та має вигляд 01JAN1960.

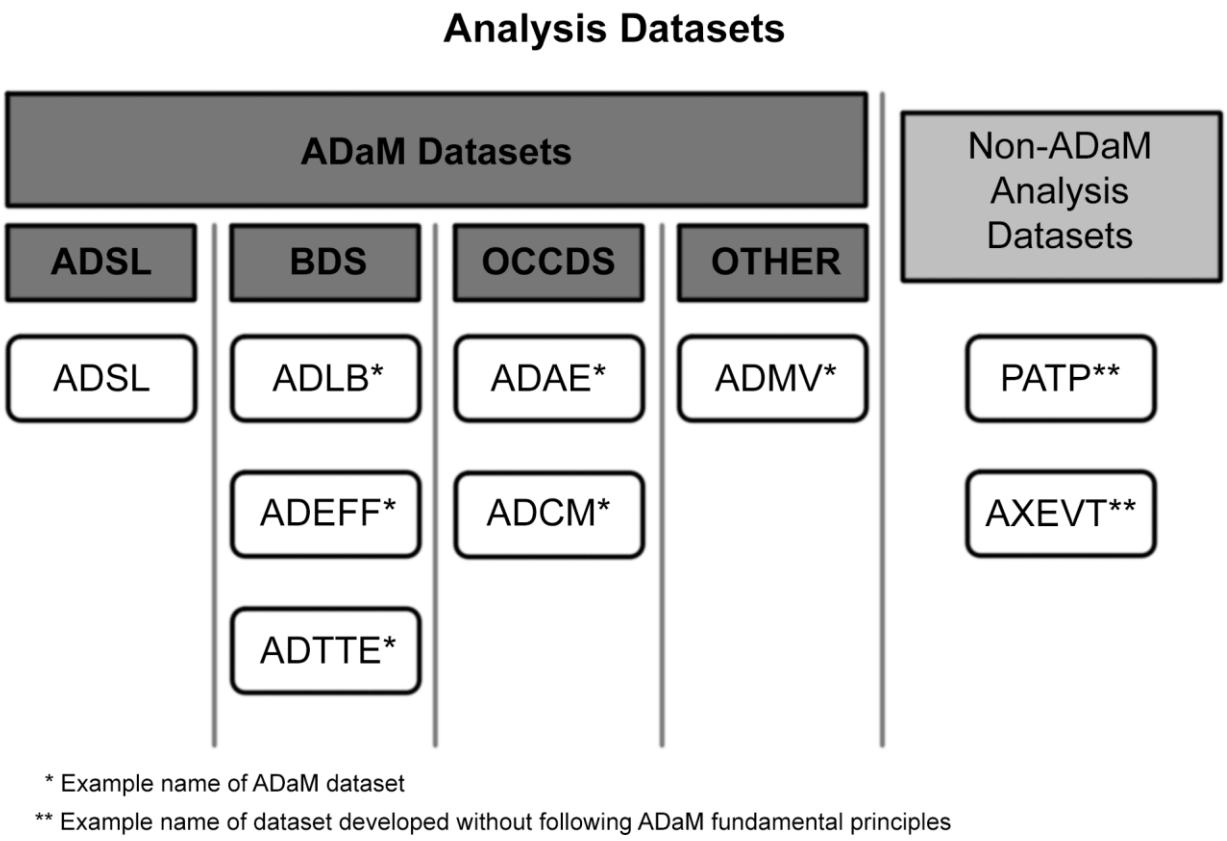

Рисунок 2 – Структури ADaM-датасетів

Безпосередньо TLF-звіти створюються відповідно до специфічної для них документації. Тобто окрім плану статистичного аналізу (SAP) існують так звані «шели» (від table shells), що являють собою макети таблиць, лістингів та фігур. Ці шели регламентують зовнішній вигляд зазначених TLF-звітів та перелічують усі необхідні вимоги до них, такі як точність представлення розрахункових величин, порядок розташування категоріальних значень тощо. Приклад процесу обробки даних наведено на рис 3.

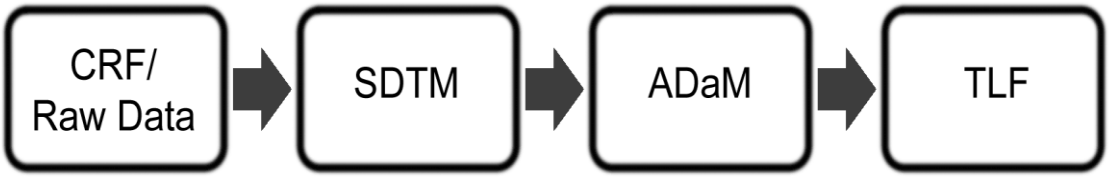

Рисунок 3 – Діаграма процесу обробки даних

Важливою особливістю обробки даних клінічних досліджень є обов'язкове дотримання концепції подвійного програмування (double programming) відповідно до директив FDA. Суть концепції полягає в тому, що на кожному з можливих етапів, будь то створення SDTM-датасетів чи TLF-звітів, програмування відбувається двічі різними програмістами. Перший програміст (first line programmer або production programmer) розробляє програмний код, що буде використовуватися для створення датасетів чи звітів та буде присутнім у пакеті даних, що відправляються до регуляторного органу разом з іншими даними та документами. Другий програміст (QC-programmer або validation programmer) розробляє свій власний код, мета якого є отримання того ж самого продукту (датасету, звіту) що і перший програміст, але на відміну від нього також виконує порівняння результатів, які було отримано обома програмістами. Результатом такої сумісної роботи двох програмістів є датасет чи звіт, що не містить помилок та виконаний із дотриманням обов'язкових стандартів та документації клінічного дослідження (протокол, SAP тощо). Є вірним вважати, що такий підхід не гарантує повної відсутності помилок у результуючих даних, але він суттєво мінімізує їхню кількість, на що значною мірою впливає компетенція програмістів.

1.6 Мета та актуальність роботи

Макропрограми відіграють життєво важливу роль у процесі автоматизації програмування SAS так як автоматизація є ключовою стратегією для підвищення ефективності та якості результатів програмування. Як результат, автоматизація за допомогою SAS-макросів або самодостатніх програм вже давно використовується як фармкомпаніями, так і CRO (Contract Research Organization).

Фармкомпанії мають на меті просту автоматизацію, бо всі роботи, що виконуються всередині такої компанії сильно стандартизовані – всі TLF-звіти будуть мати однаковий вигляд незалежно від клінічного дослідження (кількість

яких може налічувати сотні й тисячі). Автоматизація дасть змогу наймати значно менш кваліфікований персонал, бо всю складну частину робитиме розроблений компанією інструмент автоматизації.

У CRO до використовуваних макропрограм висуваються інші вимоги, що продиктовані специфікою роботи таких організацій. Клієнтами CRO є фармкомпанії, кожна з яких має власні внутрішні стандарти та вимоги щодо того, як повинні виглядати TLF-звіти. Тому CRO мають на меті не уніфікацію, а навпаки – розширення можливостей розроблюваних ними макросів. Таким чином, один макрос (або самодостатня програма) може генерувати дуже різні за змістом та виглядом звіти, щоб задовільнити вимогам всіх, в тому числі, потенційних клієнтів.

Очевидно, що внутрішні розробки як фармкомпаній так і CRO є пропрієтарними, тобто доступні тільки всередині цих компаній, а код використовуваних макросів, як і програм взагалі, є комерційною таємницею, бо забезпечує переваги на ринку. Як результат, дуже незначна кількість макропрограм є доступною у відкритих джерелах. Якщо оцінювати кількість таких програм, що присвячені саме специфіці створення TLF-звітів певної структури (хоч вона і є типовою для клінічних досліджень), кількість доступних програм зменшується значною мірою.

Серед наявних у відкритих джерелах макропрограм при детальному розборі виявляються такі недоліки, що не дозволяють їхнього адекватного впровадження. Наприклад, макрос може вимагати специфічних вхідних даних, таких як ADAE без використання ADSL. ADAE хоч і є достатньою умовою для побудови звіту по побічним ефектам, але такий макрос не має достатньої гнучкості, необхідність якої, в більшості випадків, буде продиктована вимогами конкретного клінічного дослідження [14]. На додаток, в зазначеній роботі наведено тільки функціональні частини SAS-коду, які знайомі будь-якому програмісту вище початкового рівня. Ці фрагменти коду, безумовно, є корисними, але для того, щоб вони стали інструментом автоматизації при створенні TLF-звітів необхідно додатково розробити 90% коду.

Серед функціонально цілісних, можна відзначити такі, що представлені, наприклад у [15]. Такий макрос хоча і виконує поставлену перед ним задачу створення TLF-звіту – малоймовірно буде використаний на практиці, бо майже повністю позбавлений гнучкості. Згенеровані ним результати необхідно додатково допрацьовувати для приведення до потрібного вигляду, що, зазвичай, є однією з найбільш затратних частин, як з точки зору складності, так і з точки зору затрат часу та верифікації отриманих результатів. На додаток, безпосередньо таблицю-звіт необхідно виконувати окремим кодом (наведений в зазначеній роботі).

Результати роботи більшості макросів чи самостійних програм [16], як і саму їхню роботу, складно оцінити на практиці, що є результатом відсутності кодів самих макросів та доступності для завантаження самостійних програм. Тобто з точки зору використання третіми особами, результати, що наведені в цій роботі, тільки демонструють спроможність відповідних авторів розробляти такі програми. Інакше кажучи, на практиці такі програми не відрізняються від того, що використовується всередині фармкомпаній та CRO.

Метою даної роботи є розробка засобу автоматизації процесу створення аналітичних TLF-звітів за використанням SAS Macro Facility, а саме звітів по побічним ефектам (AE, Adverse Events), супутнім медикаментам (CM, Concomitant Medications), анамнезу (MH, Medical History) та медичним процедурам (PR, Procedures).

Звіти по побічним ефектам є обов’язковою аналітичною частиною будь-якого клінічного дослідження, а їхня структура стандартизованою, тому автоматизація процесу створення таких звітів є актуальною задачею, що призведе до підвищення ефективності роботи клінічних програмістів та мінімізує кількість помилок у звітах.

Структура дослідження: робота складається зі вступу, однієї глави, загальних висновків, списку використаної літератури (19), додатків (1). Повний обсяг магістерської роботи складає 95 сторінок тексту та містить 1 таблицю і 11 рисунків.

АВТОМАТИЗАЦІЯ СТВОРЕННЯ TLF-ЗВІТІВ

2.1 Мова програмування SAS та її особливості

Найчастіше використовуваними мовами програмування в сфері клінічних досліджень є SAS, R та Pyton. Але SAS вже багато років є безумовним лідером серед них, що зумовлено двома факторами. Перший – це знайомість, за багато років застосування, ще з тих пір коли SAS був єдиною використовуваною мовою, сформувалася величина кількість фахівців, що володіють нею. Другим фактором є підхід розробників SAS до створення інструментарію: кожна з наявних процедур є валідованою, тобто результат їхнього застосування є зрозумілим та передбачуваним та не залежить від версії (зворотна сумісність). В R, наприклад, бібліотеки, що підключаються, можуть оновлюватися без відома кінцевого користувача, і ті оновлення функціоналу, що призначені для покращення роботи системи та її вдосконалення (наприклад швидкодії) можуть потенційно призводити до розбіжностей в результатах розрахунків. Для клінічних досліджень це є неприпустимим – результат не має залежати від версійності програмного забезпечення, а програм написаних багато років тому в арсеналі фармкомпаній багато. Навіть якщо фактично розбіжностей виникати не буде, в очах індустрії вони потенційно можливі саме завдяки моделі розповсюдження та оновлення. Зважаючи на багатомільярдні витрати компаній на проведення клінічних досліджень, навіть потенційна можливість такого роду ризиків є невиправданою.

SAS, раніше «Statistical Analysis System» – це розроблений Інститутом SAS набір статистичних модулів, які використовуються для обробки даних, розширеної аналітики, багатовимірного аналізу тощо [17]. В термінології SAS та клінічних досліджень кожна частина інформації називається змінною (фактично являє собою стовпчик двомірної таблиці-датасету), а кожен рядок – спостереженням.

Програмний пакет SAS складається з близько 200 компонентів. Нижче наведений перелік деяких із них [018, 019].

- SAS/STAT – Statistical analysis
- SAS/GRAPH – Graphics and presentation
- SAS/OR – Operations research
- SAS/ETS – Econometrics and Time Series Analysis
- SAS/IML – Interactive matrix language
- SAS/AF – Applications facility
- SAS/QC – Quality control
- SAS/INSIGHT – Data mining
- SAS/PH – Clinical trial analysis
- SAS Enterprise Miner – data mining
- SAS Enterprise Guide – GUI-based code editor & project manager
- SAS Enterprise BI – Suite of Business Intelligence Applications
- SAS Grid Manager – Manager of SAS grid computing environment
- SAS Customer Intelligence 360 – Customer intelligence

Програми SAS здебільшого складаються з послідовності викликів DATA- та PROC-кроків. DATA-крок призначений для маніпуляцій з даними, такими як зчитування, запис та обчислення тощо. PROC-крок, або процедурний крок, застосовується для створення звітів, обчислень, в тому числі статистичних, побудови фігур (графіків), сортування даних датасетів, їхнього транспонування тощо.

SAS містить тільки два типи змінних: символьні та числові. Числові змінні мають розмір від 3 до 8 байт, розмір символьних змінних – від 1 до приблизно 32000 байт. У випадках коли довжина чи тип змінної не задані явно, SAS визначить тип змінної та призначить її довжину автоматично у відповідності до типу та довжини записуваного до змінної значення, яке було повернуте функцією або операцією. Під операцію в даному випадку мається на увазі все окрім прямого виклику функції. Наприклад, при виклику VAR1 = cat("ABC", "123") змінній VAR1 буде призначено символьний тип, а довжину – 200 байт

(довжина може відрізнятися для різних версій SAS, але для більшості функцій у всіх версіях SAS це буде 200 байт); при виклику VAR2 = min(8, 19) змінній VAR2 буде призначено числовий тип та довжину 8 байт.

Читання датасетів та, відповідно, їхня обробка відбувається зверху-донизу, один рядок (запис датасету) за раз. Тобто доступу до інших записів датасету немає. Значення та метадані кожної змінної прочитаного запису записуються до спеціального буферу – PDV (Program Data Vector), вміст якого автоматично очищується (окрім метаданих) перед кожним читанням наступного запису вхідного датасету. Таким чином, DATA-крок фактично являє собою DO-цикл, де лічильником циклу є номер запису датасету.

Є важливим відзначити обмеження, що накладаються на назви змінних датасетів. Їхня назва має починатися з символу або нижнього підкреслення («_»), а довжина не повинна перевищувати 32 символи. Втім, ці обмеження можна відключити спеціальною опцією (validvarname = any), але на практиці обробки даних клінічних досліджень такий підхід використовується вкрай рідко, зазвичай для читання зовнішніх файлів, таких як CSV-файли, що в першому рядку мають назви змінних, які не відповідають вищезазначеним правилам SAS.

Виконання коду програм в SAS відбувається в декілька етапів. Це етап обробки макропроцесору, етап компіляції та етап виконання. На етапі компіляції відбувається перевірка синтаксису та семантичний аналіз коду. Автоматичне призначення типів та довжин змінних, що описане раніше, також відбувається на цьому етапі. Тільки у випадку вдалого проходження етапу компіляції буде виконано перехід до виконання коду програми – етапу виконання. Окремо слід відзначити етап роботи макропроцесору. Макроси в системі SAS мають свої оператори (виклик починається з знаку відсотків, наприклад %let) та свої змінні – макрозмінні, що зберігаються в окремому просторі пам'яті, так званій таблиці символів – локальній чи глобальній, від чого залежить область видимості (scope) цих змінних. Макрозмінні завжди символьні та можуть містити цілі фрагменти основного коду. В той час коли здійснюється виклик макрозмінної, макропроцесор заміняє цю змінну її вмістом, тим самим модифікуючи код

основної програми (рис. 4), що буде виконаний на етапі виконання. Таким чином, макропроцесор SAS є потужним інструментом процедурної генерації та модифікації коду виконуваної програми, завдяки чому SAS-програми в цілому, за умови використання в них макросів, є дуже гнучкими.

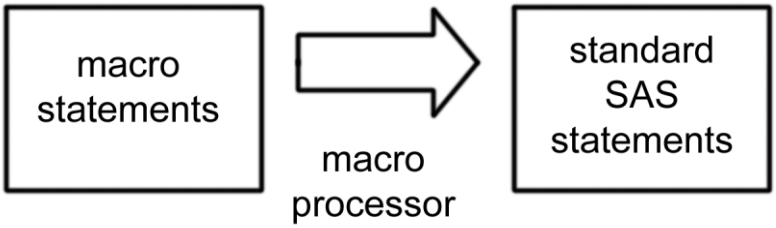

Рисунок 4 – Робота макропроцесора SAS

2.2 Постановка задачі

З моменту створення стандартів CDISC компанії, що займаються клінічними дослідженнями чи обробкою даних намагаються (CRO) автоматизувати процес програмування даних для клінічних досліджень. Так як обробка даних займає багато часу та зусиль, автоматизація є актуальною для будь-якого клінічного досліджування, отже всі компанії створюють свої власні системи з автоматизації процесів обробки даних.

З практичної точки зори, не зважаючи на те, що на даний момент вже створено безліч, як самостійних платформ автоматизації (комплексні програмні продукти), так і макропрограм на основі SAS Macro Facility, вони є недоступними зовні компанії-власника, що, безумовно, є одним з найважливіших обмежувальних факторів їхнього використання.

Не менш важливим є той факт, що зазначені платформи автоматизації, зазвичай, намагаються охопити всі аспекти обробки даних клінічних досліджень до переліку яких входять анотування CRF, створення SDTM- та ADaM-датасетів, їхніх специфікацій, аналітичних TLF-звітів, а також засобів валідації вірності виконання перерахованих процесів. Як результат – клінічний програміст вимушений запам'ятовувати безліч параметрів та налаштувань, якими він,

скоріш за все, ніколи не скористається за прямим призначенням: присутність жорстких зв'язків між параметрами дуже часто змушує використовувати dummy-значення, тобто значення, що не впливають на конкретно виконувану задачу, але потрібні бути наявні. Цей аспект суттєво піднімає поріг вимог нових програмістів – навіть експерти з багаторічним стажем роботи в індустрії технічно не можуть ефективно використовувати такі внутрішні інструменти автоматизації, бо не мали змоги з ними ознайомитися зовні компанії-власника.

Використання такого роду інструментів також призводить до втрати навичок програмування експертними фахівцями – робота зводиться до вводу потрібних значень параметрів або налаштувань платформи без розуміння процесів, що відбуваються всередині. На додаток, при виникненні некоректної роботи або виконання задач, що хоча б трохи виходять за межі можливостей платформи, програмісти вимушені звертатися до розробників відповідного програмного забезпечення або шукати шляхи обходу. Останнє майже завжди напряму заборонено компанією.

Засоби автоматизації, що є доступними у відкритих джерелах, дуже рідко відповідають вимогам, їхнім основним призначенням є вирішення дуже вузької задачі. Оскільки ці засоби автоматизації розроблені багатьма різними авторами, вони мають відмінну структуру та реалізацію, в результаті чого є значно простішим розробити новий засіб чим погодити безпомилкове виконання кількох, що є доступними, чи додати до них певний функціонал.

Зважаючи на вищесказане, задачею є розробка самодостатнього SAS-макросу відносно вузького спрямування, а саме макросу, який на базі SDTM- чи ADaM-датасетів створюватиме TLF-звіти певної структури. Такий підхід дасть змогу відмовитися від використання корпоративних рішень (де це можливо) та надасть можливість використання розробленого засобу автоматизації в будь-якій компанії та будь-яким програмістом, включаючи студентів різних профілей, які так або інакше пов'язані з фарміндустрією.

Перелік функціональних особливостей, що впроваджено в розробленому макросі:

- розрахунок частот (кількості) суб'єктів клінічного дослідження, що мали події, які відповідають певним вимогам;
- розрахунок відсоткового співвідношення суб'єктів, що мали подію відносно кількості всіх суб'єктів вибірки;
- розрахунок частот (кількості) подій, що спостерігалися у суб'єктів впродовж дослідження, які відповідають певним вимогам;
- категоризація розрахункових даних в межах певної групи;
- стратифікація результатів розрахунків за групами, до яких фактично чи планово належали суб'єкти впродовж дослідження (препарат, що приймали);
- узагальнення результатів стратифікованих за групами та виведення результату окремим стовпчиком;
- можливість обробляти та категоризувати різні за призначенням домени, серед яких побічні події, приймання супутніх медикаментів, минулі медичні події (анамнез) та процедури, які проходив суб'єкт як під час клінічного дослідження так і до нього;
- дворівневе сортування вихідних даних за потрібною стратифікаційною групою;
- обробка незакодованих вхідних даних та їхнє об'єднання всередині однієї категорії з можливістю представлення цієї категорії в самому кінці звіту;
- змінення точності представлення та форматування результатів розрахунків;
- створення окремого файлу з таблицею-звітом у вигляді RTF чи PDF;
- створення SAS-датасету з результатами розрахунків з метою їхнього зовнішнього використання.

2.3 Макрос SOC_PT для створення TLF-звітів

Для створення TLF таблиць-звітів можуть використовуватись як SDTM- так і ADaM-датасети. В даному випадку домени, які будуть вказані в описі роботи макросу є загальними назвами.

Макрос SOC_PT призначений для розрахунку кількості (частоти) суб'єктів клінічного дослідження, які мали побічні ефекти (AE – Adverse Events), приймали супутні медикаменти (CM – Concomitant Medications), мали певні записи в анамнезі (MH – Medical History) чи проходили медичні процедури (PR – Procedures), з можливістю стратифікації за досліджуваною групою (наприклад, плацебо чи досліджуваний лікарський засіб). На додаток до кількості (частоти) також розраховується відсоток цих суб’єктів всередині заданої вибірки (популяції). В кінцевому результаті створюється таблиця-звіт (TLF), куди виводяться усі розрахунки.

Категоризація результатів розрахунку виконується за SOC (System Organ Class) для аналізу побічних ефектів та анамнезу, ATC (Anatomical Therapeutic Chemical) для супутніх медикаментів, категорії для процедур.

Всередині SOC чи ATC дані аналогічним чином додатково категоризуються за PT (Preferred Term), або за відсутності PT – термом (оригінально-зібраним в CRF значенням). Приклад ADaM-датасету, що містить необхідні змінні наведено на рис 5.

Перш ніж продовжити опис макросу, необхідно зазначити що собою являють вищевказані SOC, ATC, PT та терм. Термом є назва побічної дії, препарату, тощо, в тому вигляді, як її було зібрано (записано до CRF). Значення термів зазвичай мають довільну форму та можуть бути внесені різними мовами, якщо клінічне дослідження є міжнародним. Зазвичай, збирається як назва, наприклад, препарату, варіацій якого безліч. Фактично, це торгові марки, а не назва діючої речовини цих препаратів. Очевидно, що аналізувати дані в такому вигляді недоцільно. Тому зібрані терми піддають так званому кодуванню. Кодування має на меті згрупувати препарати, якщо мова йде про супутні

медикаменти, за певними категоріями діючих речовин. Використовують для цього спеціальні словники, такі як WHODD (World Health Organization Drug Dictionary) чи MedDRA (Medical Dictionary for Regulatory Activities). В результаті кодування, всі назви лікарських засобів та їхніх варіацій (мова, одруківки, тощо) будуть мати одну стандартизовану назву. Отримані таким чином дані і знаходяться у зазначених змінних SOC, ATC, PT і тому подібних.

	STUDYID	USUBJID	AGE	SEX	RACE	TRT01A	SAFFL	AETERM	AEDECOD	AEBODSYS	TRTEMFL
1	ABC123	ABC123.00001	66	M	WHITE	Treatment A	Y	ABDOMINAL CRAMPS	Abdominal pain	Gastrointestinal disorders	Y
2	ABC123	ABC123.00001	66	M	WHITE	Treatment A	Y	DIARRHEA	Diarrhoea	Gastrointestinal disorders	Y
3	ABC123	ABC123.00001	66	M	WHITE	Treatment A	Y	RESTART OF CHRONIC DIVERTICULITIS	Diverticulitis	Infections and infestations	Y
4	ABC123	ABC123.00001	66	M	WHITE	Treatment A	Y	FEBRILE NEUTROPENIA	Febrile neutropenia	Blood and lymphatic system disorders	Y
5	ABC123	ABC123.00001	66	M	WHITE	Treatment A	Y	NAUSEA	Nausea	Gastrointestinal disorders	Y
6	ABC123	ABC123.00001	66	M	WHITE	Treatment A	Y	PYELONEPHRITIS	Pyelonephritis	Infections and infestations	Y
7	ABC123	ABC123.00001	66	M	WHITE	Treatment A	Y	FEVER	Pyrexia	General disorders and administration site conditions	Y
8	ABC123	ABC123.00001	66	M	WHITE	Treatment A	Y	THROMBOCYTOPENIA	Thrombocytopenia	Blood and lymphatic system disorders	Y
9	ABC123	ABC123.00002	76	M	WHITE	Placebo	Y	HEMORRHAGIC DIATHESIS	Haemorrhagic diathesis	Blood and lymphatic system disorders	Y
10	ABC123	ABC123.00002	76	M	WHITE	Placebo	Y	HYPERGLYCEMIA	Hyperglycaemia	Metabolism and nutrition disorders	Y
11	ABC123	ABC123.00002	76	M	WHITE	Placebo	Y	HYPOPHOSPHATEMIA	Hypophosphataemia	Metabolism and nutrition disorders	Y
12	ABC123	ABC123.00002	76	M	WHITE	Placebo	Y	NAUSEA	Nausea	Gastrointestinal disorders	Y
13	ABC123	ABC123.00002	76	M	WHITE	Placebo	Y	THROMBOSIS LEFT SUBCLAVIA VEIN	Subclavian vein thrombosis	Vascular disorders	Y
14	ABC123	ABC123.00002	76	M	WHITE	Placebo	Y	THROMBOCYTOPENIA	Thrombocytopenia	Blood and lymphatic system disorders	Y
15	ABC123	ABC123.00002	76	M	WHITE	Placebo	Y	THROMBOCYTOPENIA	Thrombocytopenia	Blood and lymphatic system disorders	Y

Рисунок 5 – Приклад фрагменту ADaM-датасету ADAE (Adverse Event Analysis Dataset)

Результатом виконання макропрограми, що описується, є два вихідних датасети та звіт у форматі RTF чи PDF (рис. 6), що містить саму таблицю з результатами розрахунків. Обидва вихідні датасети мають однакові дані, а відмінність між ними полягає в тому, що один з них матиме додаткові змінні (рис. 6а), призначенням яких є сортування (зазвичай необхідні для коректної роботи SAS-процедури proc report).

Окрім розрахунку частот суб'єктів, макрос здатен також рахувати кількість подій, що відбулися з суб'єктом, наприклад, побічних ефектів. Різниця в розрахунку полягає в наступному. Якщо розраховується частота (кількість) суб'єктів, що мали якусь подію певної категорії (SOC чи PT), суб'єкт рахується тільки один раз незалежно від того скільки однакових (або тих, що відносяться до тієї ж самої категорії) подій в нього було. В той час, як при розрахунку

кількості подій враховуються всі події. Саме тому кількість подій не супроводжуються відсотками – якщо, наприклад, у вибірці десять суб'єктів і кожен з них мав по дві події, відсоток суб'єктів з подією буде більшим за сотню. Вочевидь, репрезентативність таких результатів щонайменше сумнівна.

	PAGE_ORD	CAT_ORD	PT_ORD	CAT_PT	Stratum_A_SUBJECTS	Stratum_B_SUBJECTS	Total_SUBJECTS
1	1	1	1	Subjects with Any Adverse Event	24 (100%)	22 (100%)	46 (100%)
2	1	2	1	Gastrointestinal disorders	22 (91.7%)	18 (81.8%)	40 (87.0%)
3	1	2	2	Diarrhoea	18 (75.0%)	18 (81.8%)	36 (78.3%)
4	1	2	3	Nausea	15 (62.5%)	9 (40.9%)	24 (52.2%)
5	1	2	4	Vomiting	6 (25.0%)	10 (45.5%)	16 (34.8%)
6	1	2	5	Constipation	5 (20.8%)	4 (18.2%)	9 (19.6%)
7	1	2	6	Abdominal pain	2 (8.3%)	4 (18.2%)	6 (13.0%)
8	1	3	1	General disorders and administration site conditions	19 (79.2%)	18 (81.8%)	37 (80.4%)
9	1	3	2	Pyrexia	12 (50.0%)	11 (50.0%)	23 (50.0%)
10	1	3	3	Fatigue	7 (29.2%)	9 (40.9%)	16 (34.8%)
11	1	3	4	Pain	3 (12.5%)	4 (18.2%)	7 (15.2%)
12	1	3	5	Asthenia	2 (8.3%)	4 (18.2%)	6 (13.0%)
13	1	3	6	Oedema peripheral	3 (12.5%)	1 (4.5%)	4 (8.7%)
14	1	3	7	Non-cardiac chest pain	1 (4.2%)	2 (9.1%)	3 (6.5%)
15	1	3	8	Malaise	0	2 (9.1%)	2 (4.3%)
16	1	3	9	Impaired healing	1 (4.2%)	0	1 (2.2%)
17	1	3	10	Influenza like illness	0	1 (4.5%)	1 (2.2%)
18	2	3	11	Vessel puncture site haematoma	1 (4.2%)	0	1 (2.2%)

а)

	CAT_PT	Stratum_A_SUBJECTS	Stratum_B_SUBJECTS	Total_SUBJECTS
1	Subjects with Any Adverse Event	24 (100%)	22 (100%)	46 (100%)
2	Gastrointestinal disorders	22 (91.7%)	18 (81.8%)	40 (87.0%)
3	Diarrhoea	18 (75.0%)	18 (81.8%)	36 (78.3%)
4	Nausea	15 (62.5%)	9 (40.9%)	24 (52.2%)
5	Vomiting	6 (25.0%)	10 (45.5%)	16 (34.8%)
6	Constipation	5 (20.8%)	4 (18.2%)	9 (19.6%)
7	Abdominal pain	2 (8.3%)	4 (18.2%)	6 (13.0%)
8	General disorders and administration site conditions	19 (79.2%)	18 (81.8%)	37 (80.4%)
9	Pyrexia	12 (50.0%)	11 (50.0%)	23 (50.0%)
10	Fatigue	7 (29.2%)	9 (40.9%)	16 (34.8%)
11	Pain	3 (12.5%)	4 (18.2%)	7 (15.2%)
12	Asthenia	2 (8.3%)	4 (18.2%)	6 (13.0%)
13	Oedema peripheral	3 (12.5%)	1 (4.5%)	4 (8.7%)
14	Non-cardiac chest pain	1 (4.2%)	2 (9.1%)	3 (6.5%)
15	Malaise	0	2 (9.1%)	2 (4.3%)
16	Impaired healing	1 (4.2%)	0	1 (2.2%)
17	Influenza like illness	0	1 (4.5%)	1 (2.2%)
18	Vessel puncture site haematoma	1 (4.2%)	0	1 (2.2%)

б)

Рисунок 6 – Приклад фрагментів вихідних датасетів, що згенеровані SOC_PT макросом, де а) − вихідний датасет з сортувальними змінними (PAGE_ORD, CAT_ORD, PT_ORD) та б) − вихідний датасет без сортувальних змінних

Макрос підтримує певну кількість модифікацій вигляду отримуваного результату – форми в якій буде представлено цей результат. Це налаштовується за допомогою вхідних параметрів макросу, опис яких буде наведено нижче.

Для клінічних досліджень, що є не завершеними (тобто тих, які в процесі збору даних) дуже часто виникає ситуація, коли доступним для обробки є тільки терм – кодування «свіжих» даних ще не відбулося. В такому випадку, всі незакодовані події можуть бути відображені в таблиці під категоріями «Not Coded», «Uncoded» і тому подібних, а замість значень PT використовують зібрані терми. В макрос додано такий функціонал: через значення параметрів можна змінити зазначений текст та перемістити цю категорію в кінець таблиці, незалежно від потрібного сортування.

Якщо дані для аналізу з якоїсь причини відсутні, макрос створить пустий вихідний датасет та пусту таблицю-звіт, в якій будуть наявні всі типові данні, на кшталт назв, футноутів та колонок з назвами, але в місці де повинні бути результати – буде виведено текстову строку, яка свідчить про відсутність даних для аналізу. Текст можна редагувати за допомогою одного з параметрів макросу. Приклад таблиці-звіту, що згенерований на базі відсутніх даних наведено на рис. 7, виклик макросу здійснено зі значенням параметру nodata_string_text = No Data to Report.

Результати розрахунку, що попадають у таблицю, за замовчуванням сортуються за колонкою «Total», але це можна змінити відповідним параметром. При цьому першим рядком в таблиці завжди буде та, що показує кількість всіх подій – незалежно від категорій. Після цього, за зменшенням частот розташовуються категорії (SOC, ATC), а під ними аналогічним чином відсортовані значення PT. У випадках коли декілька SOC чи PT мають однакові частоти, вони додатково сортуються за алфавітом. Категорія, що містить незакодовані події, як зазначалося вище, знаходиться або в кінці таблиці, або підкоряється правилам, які є дійсними для закодованих значень.

Макрос також має можливість змінювати порядок колонок в таблиці, що відповідають назвам груп стратифікації. Але колонка «Total» буде завжди останньою.

Table 1.1
Summary of Adverse Events by System Organ Class and Preferred Term

Population: Safety

System Organ Class Preferred Term	Stratum A N=24 n (%)	Stratum B N=22 n (%)	Total N=46 n (%)

No Data to Report

Note: Percentages are based on the number of subjects in safety population.
Subjects are counted once within each system organ class and each preferred term.
All investigator adverse event terms were coded using MedDRA dictionary version 26.0.

Рисунок 7 – Приклад таблиці, що генерує макрос SOC_PT за умови відсутності даних у вхідному датасеті

2.4 Опис функціональних можливостей макросу

Загалом макрос налічує 28 параметрів, більшість з яких є обов'язковими. Перші два параметри – dsin_SL та dsin_DATA визначають назву та розташування (бібліотеку) вхідних датасетів для розрахунків. В першому параметрі повинно бути посилання на ADSL (Subject-Level Analysis Dataset – ADaM що містить дані рівня суб'єкту, такі як популяційні флаги, стратифікаційні групи, тощо), а в

другому – посилання на один з чотирьох можливих датасетів з даними, які будуть аналізуватися.

Параметр domain є перемикачем типу датасету (допустимі значення: AE, CM, MH, PR), тобто вказує на тип датасету, що зазначений у параметрі dsin_DATA. Можливі вхідні датасети мають різні назви змінних, що підлягають обробці. За значенням цього параметру макрос автоматично визначає назви потрібних змінних. Наприклад, для побічних ефектів (AE) цими змінними будуть AEDECOD та AETERM, а для супутніх медикаментів (CM) – CMDECOD та CMTRT.

Через значення параметру dsout задається ім'я та бібліотека вихідного датасету, в якому міститимуться дані розрахунків, за замовчуванням це soc_pt_out.

Параметри trt_var та trt_varn відповідають стратифікаційним групам для розрахунку частот, що відбивається на кількості та назвах стовпчиків в самій таблиці. Змінна, що вказана у trt_var має бути присутньою хоча б в одному з вхідних датасетів, а значення за замовчуванням – TRT01A (змінна, в якій міститься значення-назва лікувального препарату, який приймає суб’єкт). Значення параметру trt_varn є опціональним, зазвичай це числовий еквівалент змінної, що передана у параметрі trt_var (наприклад, TRT01AN). Значенням змінних, що передані через параметри trt_var та trt_varn, повинні мати співвідношення один-до-одного. Таким чином, якщо параметр trt_varn не пустий, стовпчики стратифікаційних груп та змінні у вихідному датасеті буде розташовано відповідно до числових значень змінної, що передано у trt_varn (зліва-направо). У протилежному випадку (параметр пустий), розташування також буде зліва-направо, але в алфавітному порядку.

Значення за якими категоризуються частоти визначається параметром cat_var. Для побічних ефектів (AE) це зазвичай це AESOC. Значення самої першої категорії, в якій розраховано кількість подій на суб'єкта без категоризації, завжди знаходиться вгорі таблиці, а назву (текст, що виводиться) для цієї

категорії змінюється шляхом оновлення значення параметра any_string_text. За замовчуванням значення дорівнює «Subjects with Any Adverse Event» (без лапок).

Фільтрування вхідних даних відбувається за використанням параметрів pop_var та filter. В параметрі pop_var вказується ім'я популяційного флагу (вибірки) для аналізу, ця змінна обов'язково присутня у ADSL, ім'я та розташування якого передано через параметр dsin_SL. Другий параметр, filter, – це фільтр, який накладається через оператор where після об'єднання вхідних датасетів. Тобто, розрахунок буде виконуватися з урахуванням тільки тих записів, що відповідають умові зазначеній в параметрі filter, та які мають значення «Y» у змінній популяції, яку передано через pop_var.

Параметр do_calc_evt є тригером необхідності виконання додаткового розрахунку кількості (частот) подій. За замовчуванням, має значення false, і в такому випадку рахується кількість унікальних подій, що були у суб'єкта.

Параметри do_calc_total та total_col_name відповідають наявності стратифікаційної групи «Total» (або «Overall»). Фактично, значення частот відповідного стовпчика у таблиці є сумою частот всіх інших наявних груп. За замовчуванням значення параметру do_calc_total є true, тобто розрахунок буде виконано. Для випадків, коли група всього одна, або просто немає необхідності в групі «Total», для параметру do_calc_total необхідно встановити значення false. Значення, що міститься у параметрі total_col_name впливає на текст в шапці стовпчику, що відповідає групі «Total», а також на назву відповідної змінної у вихідному датасеті. Значення за замовчуванням – «Total».

Дані в таблиці сортуються за зменшенням частот для наявних категорій (SOC, ATC). Значення PT сортуються аналогічним чином, але в середині категорії, до якої вони належать. У випадку рівності частот категорій або термів всередині цих категорій, додаткове сортування відбувається за алфавітом. Є очевидним, що результати розрахунку будуть різними для різних стратифікаційних груп (власне для порівняння такі розрахунки і робляться), тому послідовність даних у таблиці (та вихідному датасеті) буде залежати від того, дані якої з груп використані для сортування. У більшості випадків, сортування

виконується за групою «Total», але, якщо групу потрібно змінити – необхідно оновити значення параметру out_sort_by. За замовчуванням цей параметр має значення «Total», а вхідне значення «Overall» інтерпретує, як ідентичне значенню «Total». Допустимими значеннями для параметру out_sort_by є ті значення, що знаходяться у змінній, яку передано через параметр trt_var.

До логіки відображення вихідних даних, що описана вище, додано можливість виключення для незакодованих значень. Часто, вимогою є те, щоб категорія, на кшталт, «Not coded», та відповідні їй терми, розташовувалися у низу таблиці. Для цього потрібно передати значення true параметру nc_put_last, яке за замовчуванням дорівнює false. Сама назва категорії задається через параметр nc_string_text, значення за замовчуванням якого «Not coded» (без лапок).

Текст, що виводиться в шапці головної колонки з даними, в якій розташовано назви категорій та підкатегорій, для яких пораховано частоти, залежить від типу оброблюваних даних (побічні події, супутні медикаменти, тощо), а отже необхідно передбачити можливість його зміни. Це можна виконати за допомогою параметру cat_header_text, значення якого, за замовчуванням, «System Organ Class /Preferred Term» (без лапок).

Текст повідомлення про відсутність вхідних для обробки можна оновити через значення параметра nodata_string_text. Значення, що виводиться за замовчуванням – «*****No Data Available for Processing******» (без лапок).

Параметри decimals та out_aligned відповідають за вигляд даних у таблиці та вихідному датасеті. Перший параметр визначає кількість знаків після крапки, які треба виводити для відсотків. Наприклад, якщо розрахункове значення відсотків дорівнює 85,2638, то при decimals = 1 (значення за замовчуванням) відсотки будуть мати вигляд 85,3%, а при значенні decimals = 3 – 85,264%. Параметр out_aligned, в свою чергу, відповідає за горизонтальне вирівнювання результатів у колонках таблиці. Так, при значенні out_aligned = false, яке є значенням за замовчуванням, дані буде вирівняно по лівому краю, а при значенні out_aligned = true – вирівнювання буде зроблене за шириною (рис. 8).

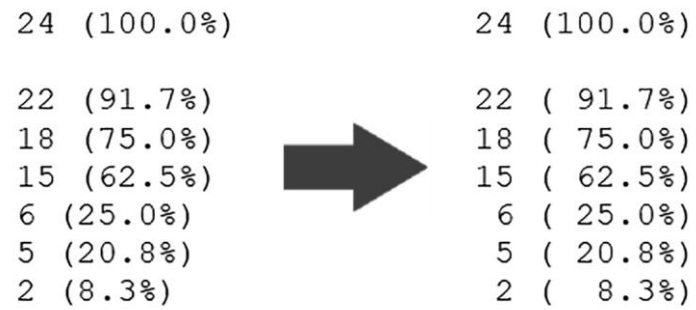

Рисунок 8 – Приклад відкоректованого вигляду значень

У випадках, коли потрібно подивитися або використати проміжні дані, що створює макрос в процесі власної роботи, параметру do_cleanup призначається значення false. В протилежному випадку, (do_cleanup = true, значення за замовчуванням) всі датасети, окрім того, що вказаний у параметрі dsout та report_dsin (використовується як вхідний для процедури proc report) будуть видалені. Інформацію, що стосується кількості суб'єктів у виборці (параметр pop_var) кожної групи (параметр trt_var), та за яким стовпчиком виконане кінцеве сортування даних (параметр out_sort_by), можна продивитися у лозі програми, якщо змінити значення параметру do_out_info з false (за замовчуванням) на true.

За допомогою параметру do_out_report можна вимкнути створення звіту у PDF чи RTF файлі, що є корисним на етапі попередньої обробки, а шляхом підбору параметрів lines_ppage та cat_pt_linelen налаштувати зовнішній вигляд таблиці, де параметр lines_ppage відповідає за кількість рядків на сторінку, що є доступними для виводу, а cat_pt_linelen вказує скільки літер (символів) можна вмістити до одного рядка головного стовпчика (з категоріями). На основі значень параметрів lines_ppage та cat_pt_linelen макрос розраховує в якому місці необхідно робити розриви сторінок. Такий підхід є вимушеним при використанні RTF-формату для виводу, у випадку, якщо вивід здійснювався б у звичайний текстовий документ з використанням моноширинних шрифтів.

Параметри, out_file та rtf_pdf, визначають місце, де потрібно зберегти файл звіту, та його тип. Можливі значення для rtf_pdf, як можна зрозуміти з назви параметру, є RTF чи PDF, а значенням за замовчуванням – RTF.

Значенням параметру simple_varnames (за замовчуванням false) обирається метод генерації імен змінних, що містять результати розрахунку частот. При значенні false змінні будуть мати назви відповідно до назв стратифікаційних груп, а у протилежному випадку матимуть назви SUBJ_TRT01, SUBJ_TRT02.

2.5 Опис роботи макросу

Код SAS макросу розташовується між ключовими словами %macro та %mend, де після %macro вказується ім'я макросу та перелік параметрів, що він приймає, у дужках. Параметри можуть бути іменні та позиційні. В випадку даного макросу всі параметри є іменними, це означає, що можна викликати будь-яку кількість параметрів та у будь-якому порядку. Далі будуть розглянуті ключові функціональні фрагменти коду макросу, а весь код макросу можна переглянути у додатку. Нижче наведений код ініціалізації макросу SOC_PT, де після знаку рівності вказані значення параметрів за замовчуванням.

```
%macro SOC_PT(
     dsin_SL = ADSL,
     dsin_DATA = ADAE,
     domain = AE,
     dsout =,
     trt_var = TRT01A,
     trt_varn =,
     cat_var =,
     pop_var =,
     filter =,
     do_calc_evt = false,
     do_calc_total = true,
     total_col_name = Total,
     out_sort_by = total,
```

```
decimals = 1,
out_aligned = false,
any_string_text = Subjects with Any Adverse Event,
nc_put_last = false,
nc_string_text = Not Coded,
do_cleanup = true,
do_out_info = false,
do_out_report = true,
lines_ppage = 20,
nodata_string_text = *****No Data Available for
Processing******,
cat_pt_linelen = 60,
cat_header_text = System Organ Class /Preferred Term,
out_file =,
rtf_pdf = RTF,
simple_varnames = false);
```

Першою частиною макросу є ініціалізація макрозмінних, як глобальних так і локальних. Різниця між ними полягає в тому, що локальні змінні буде видалено відразу після виконання макросу, а глобальні макрозмінні залишаться доступними для подальшого використання. До глобальних макрозмінних виведено такі змінні як trt_cnt, trt_names, trt_totals та trt_ords. Їхні значення відповідають кількості стратифікаційних груп – кількість значень змінної, що передано через параметр trt_var (плюс одна змінна, якщо параметр do_calc_total = true), перелік самих значень, кількості суб'єктів, що належать цій групі з урахуванням вибірки вказаної у параметрі pop_var, та порядок цих груп, який буде використано для визначення порядку змінних вихідного датасету (параметр dsout) та таблиці зі звітом. Останнє значення або відповідає значенням змінної, що передане через параметр trt_varn, або розраховане, якщо параметр пустий.

```
%global trt_cnt trt_names trt_totals trt_ords;

%local macro_name tot_subjs count_varname count_mode
decimals_num move_left out_sort_varname out_vars_list
cat_count length_max out_log_text simple_varnames data_rec_num
out_style hor_margin out_trt_col_width decod_varname
term_varname;
```

Всі використовувані макрозмінні макросу проходять етап перевірки та, за необхідності, реініціалізації значень за замовчуванням. Якщо макрос викликано з вказанням імені параметру але без відповідного значення, SAS вважатиме це як передачу пустого значення. У переважній кількості випадків, це заважатиме користувачеві, тому вірним підходом є виклик макросу тільки з не пустими параметрами для забезпечення збереження значень за замовчуванням.

```
%let macro_name = SOC_PT;
%let hor_margin = 0.00;
%let out_style = just = left asis = on font_face = Arial
font_size = 10 marginleft = &hor_margin.in marginright =
&hor_margin.in;

%let simple_varnames = FALSE;
%if %upcase(&out_aligned.) ^= TRUE %then %let move_left = -L;
%else %let move_left =;
%if &dsout. = %then %let dsout = soc_pt_out;
%if &decimals. = %then %let decimals = 1;
%if &lines_ppage. = %then %let lines_ppage = 40;
%if "&out_sort_by." = "" or "%upcase(&out_sort_by.)" =
"OVERALL" %then %let out_sort_by = TOTAL;
```

```
%if "&total_col_name." = "" %then %let total_col_name = Total;
%if "&any_string_text." = "" %then %let any_string_text =
Subjects with Any &domain.;
%if "&nc_string_text." = "" %then %let nc_string_text = Not
Coded;
%if "&nodata_string_text."  = "" %then %let nodata_string_text
= *****No Data Available for Processing******;
%if "&cat_header_text." = "" %then %let cat_header_text =
System Organ Class /Preferred Term;
%if "&out_file." = "" %then %let out_file = C:\soc_pt.rtf;
```

Тут перевіряються значення ключових параметрів макросу на відсутність значення, і якщо таке виявлене, параметру знову призначається значення за замовчуванням. Також відбувається ініціалізація деяких змінних значеннями за замовчуванням, в деяких випадках – в залежності від значень вхідних параметрів. Унікальними змінними є macro_name, hor_margin та out_style – вони задаються тільки всередині макросу, тобто змінити їх не змінюючи код макросу неможливо. Змінна macro_name відповідає за підстановку назви цього макросу у текстові повідомлення до логу програми, а змінні hor_margin та out_style модифікують вигляд тексту – розмір та тип шрифту, горизонтальне вирівнювання та горизонтальні відступи всередині комірок таблиці.

Як зазначалося раніше, категоризація результатів розрахунків відбувається за SOC (АТС) та PT (див. розд. 2.3), і якщо змінну для SOC-категоризації визначає значення параметру cat_var, змінні що рахується – PT та терми – визначаються всередині макросу відповідно до обраного домену (типу оброблюваних даних), вказаного в параметрі domain. Значення термів буде використане тільки у випадках, коли значення SOC та PT відсутні, тобто, кодування ще не відбулося. Імена цих змінних зберігаються до макрозмінних decod_varname та term_varname для подальшого використання.

```
%if %upcase(&domain.) = AE %then %do;
  %let decod_varname = AEDECOD;
  %let term_varname = AETERM;
%end;
%else %if %upcase(&domain.) = CM %then %do;
  %let decod_varname = CMDECOD;
  %let term_varname = CMTRT;
%end;
%else %if %upcase(&domain.) = MH %then %do;
  %let decod_varname = MHDECOD;
  %let term_varname = MHTERM;
%end;
%else %if %upcase(&domain.) = PR %then %do;
  %let decod_varname = PRDECOD;
  %let term_varname = PRTRT;
%end;
%else %do;
  %put %str(ERR)OR: Macro &macro_name. %str(err)or: The only
acceptable values for the domain parameter are AE, CM, MH or
PR. The macro will exit now.;
  %return;
%end;
```

Макрос налаштований на роботу з чотирма можливими доменами, це AE (Adverse Events, або побічні події), CM (Concomitant Medications, або супутні медикаменти), MH (Medical History, або медична історія) та PR (Procedures, або процедури). Ці домени мають однакову структуру даних SDTM-датасетів та майже завжди їхні данні необхідні для аналізу (створення відповідних таблиць). Таким чином, макрозмінні decod_varname та term_varname матимуть значення AEDECOD та AETERM для domain = AE, CMDECOD та CMTRT для

domain = CM, MHDECOD та MHTERM для domain = MH, PRDECOD та PRTRT для domain = PR. У випадку коли значення параметру domain є відмінним від цих перерахованих варіантів, макрос виведе до логу програми повідомлення про помилку (рис. 9) та припинить своє виконання.

Рисунок 9 – Приклад логу

Робота макросу не є можливою без наступних значень параметрів: назви вхідних датасетів, типу домену датасету з даними, змінної для категоризації та змінної з назвами стратифікаційних груп. Тому макрос виконує перевірку відповідних значень параметрів, а саме: dsin_SL, dsin_DATA, domain, trt_var та cat_var. У випадку відсутності хоча б одного з параметрів, макрос виводить строку з текстом про помилку до логу програми та припинить виконання. Якщо перевірка була вдалою, після неї створюється локальна копія (в бібліотеці work) датасету з даними рівня суб'єкту (ADSL, Subject-Level Analysis Dataset) та його попереднє фільтрування (так званий сабсетінг): обираються для подальшого аналізу тільки ті записи з ADSL, що відповідають необхідній вибірці для створюваної таблиці. Змінна вибірки, яка також називається популяційним флагом, передається через параметр макросу pop_var. Якщо значення цієї змінної у ADSL дорівнює «Y», то відповідний запис належить зазначеній вибірці, в протилежному випадку, коли значення дорівнює «N» – не належить. Слід зазначити, що локальна копія ADSL–датасету створюється за допомогою SAS-процедури сортування (proc sort), отже цей датасет буде мати сортування за значеннями змінних STUDYID та USUBJID, які є ключем для цього датасету.

```
%if &dsin_SL. = or &dsin_DATA. = or &domain. = or &trt_var. =
or &cat_var. = %then %do;
```

```
  %put %str(ERR)OR: Macro &macro_name. %str(err)or: All core
parameters (dsin_SL, dsin_DATA, domain, trt_var, cat_var) must
be provided. The macro will exit now.;
  %return;
%end;

proc sort data = &dsin_SL. out = adsl_data;
     by STUDYID USUBJID;
     where %if &pop_var. ^= %then &pop_var. = 'Y';;
run;
```

Стовпчики таблиці, що відповідають стратифікаційним групам, зазвичай, мають певний порядок, який може бути різним в залежності від вимог клінічного дослідження. Тому в макросі передбачена можливість відповідної зміни порядку цих стовпчиків та відповідних змінних вихідних датасетів. Для визначення порядку використовується пара значень, що міститься у змінних параметрів trt_var та trt_varn. Наприклад, змінна TRT01A (trt_var = TRT01A) має значення «Treatment Group A», «Treatment Group B», «Placebo», а змінна TRT01AN відповідно значення 1, 3 та 2. В такому випадку зазначені стовпчики та відповідні ним змінні вихідних датасетів буде розташовано в такому порядку: «Treatment Group A», «Placebo», «Treatment Group B».

Механізм сортування реалізовано за допомогою створення потрібного символьного SAS-інформату (тут його назва «ifmt_trtn»), що генерується на базі датасету зі значеннями змінних TRT01A та TRT01AN (відповідно до прикладу вище). Код, який наведено нижче генерує датасет з потрібною для створення на його базі інформату та створює сам інформат за використанням SAS-процедури proc format з опцією cntlin =. У випадку коли через параметр trt_varn не передано значення, цей код створить локальний дублікат потрібної змінної, де значення 1 буде відповідати першому за алфавітом значенню змінної у параметрі trt_var, тобто вихідні стовпчики будуть розташовані зліва-направо в алфавітному

порядку за назвою стратифікаційної групи. Для наведеного прикладу, це буде «Placebo», «Treatment Group A», «Treatment Group B».

Змінну з числовим еквівалентом стратифікаційної групи, або її порядок слідування, буде додано до тимчасової копії ADSL-датасету в бібліотеці work.

```
proc sql noprint;
  create table fmt_ds as
    select distinct 'ifmt_trtn' as FMTNAME,
      upcase(&trt_var.) as START,
      'i' as TYPE,
      'U ' as HLO
      %if &trt_varn. ^= %then %str(, &trt_varn. as LABEL);
    from adsl_data
    %if &trt_varn. = %then %str(order by START);
    %else %str(order by LABEL);
  ;
quit;

data fmt_ds;
  length LABEL 8;
  set fmt_ds end = EOF;
  %if &trt_varn. = %then %str(LABEL + 1);
  output;
  if EOF then do;
    START = upcase("&total_col_name.");
    LABEL = LABEL + 1;
    output;
  end;
run;
```

```
proc sort data = fmt_ds;
  by START;
run;

proc format cntlin = fmt_ds;
run;

%if &trt_varn. = %then %do;
  %let trt_varn = &trt_var._N;
  data adsl_data;
    length &trt_varn. 8;
    set adsl_data;
    &trt_varn. = input(&trt_var., ifmt_trtn.);
  run;
%end;
```

Перед розрахунком необхідних частот датасет ADSL та датасет для аналізу об'єднується на data-кроці за допомогою оператору merge за спільним для обох датасетів ключем – змінні STUDYID та USUBJID в операторі by. Результат поєднання буде використовуватися як база для розрахунку частот. Через опцію in = визначається присутність запису в кожному з вхідних датасетів (adsl_data та input_data). Для того, щоб запис попав до вихідного датасету (sl_input_data) потрібно щоб він був наявним в обох вхідних датасетах.

```
proc sort data = adsl_data;
  by STUDYID USUBJID;
run;
proc sort data = &dsin_DATA. out = input_data;
  by STUDYID USUBJID;
run;
```

```
data sl_input_data;
  merge adsl_data  (in = dsin_SL)
        input_data (in = dsin_DATA);
  by STUDYID USUBJID;
  if dsin_SL and dsin_DATA;
run;
```

Наступним кроком є розрахунок кількості суб'єктів, що є присутніми в кожній стратифікаційній групі з урахуванням популяції – так званих великих N-ок («big Ns»). Суб'єкт може входити до вибірки та групи але не мати подій (медикаментів, процедур, тощо), що аналізуються, тому для розрахунку N використовуються відфільтровані дані ADSL-датасету – дані по такому суб'єкту будуть обов'язково присутні в ADSL. Ці значення N виводяться поряд з назвами стратифікаційних груп в шапках відповідних стовпчиків таблиці, а також додаються до лейблів відповідних змінних вихідних датасетів (для програмної перевірки значень N шляхом порівняння метаданих вихідного датасету). Значення N також є знаменником для розрахунку кількості подій у відсотках, тобто відсотки рахуються (поки не зазначене інше) відносно значень N. Отримані дані щодо кількості наявних стратифікаційних груп, їхнього порядку, назв та значення N записуються до макрозмінних trt_cnt, trt_names, trt_totals та trt_ords.

```
proc sql noprint;
  select n(distinct &trt_var.),
         n(distinct USUBJID)
    into :trt_cnt   trimmed,
         :tot_subjs trimmed
    from adsl_data
  ;
  select distinct &trt_var.,
```

```
                  n(distinct USUBJID),
                  &trt_varn.
    into :trt_names  separated by '|',
         :trt_totals separated by '|',
         :trt_ords   separated by '|'
    from adsl_data
    group by &trt_var.
    order by &trt_varn.
  ;
quit;
```

Генерація назв змінних стратифікаційних груп відповідно до вимог SAS (наприклад, назва не може починатися з числа) виконано за допомогою подвійного транспонування, як показано нижче. Згенеровані назви записуються до макрозмінних. Отримані таким чином назви буде використано пізніше за умови значення false парметру simple_varnames. Кінцеві значення, з урахуванням групи «Total» (її розташування, значення N, тощо) зберігаються до макрозмінних trt_cnt, trt_names, trt_vnames, trt_totals, trt_ords та trt_vords.

```
proc sort data = adsl_data out = trtgrp_names (keep =
&trt_var. &trt_varn.) nodupkey;
  by &trt_varn.;
run;

proc transpose data = trtgrp_names out = trtgrp_names_tr (drop
= _:);
  var &trt_varn.;
  id &trt_var.;
run;
```

```
proc transpose data = trtgrp_names_tr out = trtgrp_vnames
(rename = (COL1 = &trt_varn.)) NAME = &trt_var._VNAME;
  var _all_;
run;

proc sql noprint;
  select distinct &trt_var._VNAME, &trt_varn.
    into :trt_vnames separated by '|',
         :trt_vords  separated by '|'
    from trtgrp_vnames
    order by &trt_varn.
  ;
quit;

%let trt_cnt = %eval(&trt_cnt. + 1);
%let trt_names = &trt_names.|&total_col_name.;
%let trt_vnames = &trt_vnames.|&total_col_name.;
%let trt_totals = &trt_totals.|&tot_subjs.;
%let trt_ords = &trt_ords.|%sysfunc(inputn(&total_col_name.,
ifmt_trtn.));
%let trt_vords = &trt_vords.|%sysfunc(inputn(&total_col_name.,
ifmt_trtn.));
```

До датасету, що містить дані для обробки додаються дані для «Total» групи. Фактично це копіювання всіх наявних даних всіх інших стратифікаційних груп з заміною значень відповідних змінних (що записані у trt_var та trt_varn). Такий підхід дає змогу рахувати цю «Total» групу таким самим чином, як і інші без додаткових обробок: технічно просто з'являється додаткова стратифікаційна група у вхідних даних.

```
data sl_input_data;
  set sl_input_data;
  output;
  &trt_var. = "&total_col_name.";
  &trt_varn. = input(&trt_var., ifmt_trtn.);
  output;
run;
```

Оскільки, зазвичай, підрахунок виконується не для всіх наявних даних, а тільки для тих, що відповідають певним вимогам, комбінований датасет з даними фільтрується відповідно до значення параметрів filter та pop_var. Прикладом таких вимог може бути умова, що побічна дія відбулася в певний проміжок часу, такий як між датою прийому першої та останньої доз досліджуваного лікарського засобу.

```
data sl_input_data;
  set sl_input_data;
  where
    %if &pop_var. ^= %then &pop_var. = 'Y';
    %if &pop_var. ^= and &filter. ^= %then %str( and );
    %if &filter. ^= %then &filter.;
  ;
run;
```

В процесі обробки даних може виявитися, що жоден з записів вхідного датасету не відповідає необхідним умовам. Відповідно до прикладу вище, самі АЕ-події у суб'єктів є, але жодна не відбулася впродовж зазначеного проміжку часу. В таких випадках, виводять пустий датасет та таблицю, в якій відсутня частина з результатами розрахунків. Тобто в таблиці зберігається присутність назви, футноутів, стовпчиків з назвами стратифікаційних груп та відповідних до

них значень N, але в місці де мають бути дані виводиться текст, що вказує на відсутність вхідних даних для розрахунку. Сам текст такого повідомлення задається через значення параметру nodata_string_text, а його значення за замовчуванням – «*****No Data Available for Processing******» (без лапок).

З точки зору роботи макросу, це означає, що певну частину розрахунків потрібно пропустити та перейти до формування пустої таблиці та вихідного датасету. В коді даного макросу такий перехід виконано за допомогою макрооператору %goto. Відмітка для переходу - skip_calc. Відсутність даних для обробки визначається на основі кількості записів датасету з даними, ця кількість записується до макрозмінної data_rec_num використовуючи data-крок.

```
data _null_;
  call symputx("data_rec_num", OBS_CNT);
  if 0 then set sl_input_data nobs = OBS_CNT;
run;

%if &data_rec_num. = 0 %then %goto skip_calc;
```

Розрахунок частот виконується за використанням процедури proc sql всередині циклу як показано нижче. Якщо потрібен тільки розрахунок частот суб'єктів, то обробку всередині циклу буде виконано лише один раз, а у випадку коли окрім частот суб'єктів необхідно також порахувати кількість подій – код буде виконано двічі. Частоти для кожної категорії та даних всередині цих категорій рахуються окремими запитами, а потім результати розрахунків об’єднуються в одному датасеті за допомогою outer union corr. Таким чином, буде отримано один чи два датасети – один з кількістю суб'єктів (SUBJ_COUNT), а другий з кількістю подій (EVNT_COUNT). На даному етапі також відбувається підстановка тексту назви категорії за відсутності кодування змінних, та використовується терм замість PT.

```
%if &i. = 1 %then %do;
  %let count_varname = SUBJ_COUNT;
  %let count_mode = %str(distinct);
%end;

%else %do;
  %let count_varname = EVNT_COUNT;
  %let count_mode  = %str();
%end;

proc sql noprint;
  create table &count_varname. as
    select n(&count_mode. USUBJID) as &count_varname.,
&trt_var., &trt_varn., 'ANY' as CAT
    from sl_input_data
    group by &trt_var., &trt_varn.
  outer union corr
    select n(&count_mode. USUBJID) as &count_varname.,
&trt_var., &trt_varn., &cat_var., 'CAT_VAR' as CAT
    from sl_input_data
    where ^missing(&cat_var.)
    group by &trt_var., &trt_varn., &cat_var.
  outer union corr
    select n(&count_mode. USUBJID) as &count_varname.,
&trt_var., &trt_varn., "&nc_string_text." as &cat_var.,
'CAT_VAR' as CAT
    from sl_input_data
    where missing(&cat_var.)
    group by &tr_var., &trt_varn., &cat_var.
```

```
  outer union corr
    select n(&count_mode. USUBJID) as &count_varname.,
&trt_var., &trt_varn., &cat_var., &decod_varname., 'DECOD' as
CAT
    from sl_input_data
    where ^missing(&cat_var.)
    group by &trt_var., &trt_varn., &cat_var., &decod_varname.
  outer union corr
    select n(&count_mode. USUBJID) as &count_varname.,
&trt_var., &trt_varn., "&nc_string_text." as &cat_var.,
&term_varname., 'TERM' as CAT
    from sl_input_data
    where missing(&cat_var.)
    group by &trt_var., &trt_varn., &cat_var., &term_varname.
    order by &trt_varn., &cat_var., CAT, &decod_varname.,
&term_varname.
  ;
quit;
```

Розрахунок максимальної «ширини» значення частот також виконується за допомогою proc sql, а результат записується до макрозмінної, яка має таку ж саму назву як і датасет, що містить пораховані частоти. Під шириною в даному випадку мається на увазі кількість символів, яку займає значення. Логіка розрахунку цієї ширини наступна. Спочатку знаходиться саме велике значення, воно перетворюється з числового на символьне, а потім рахується кількість символів. Отримана ширина буде використана пізніше для налаштування формату відображення вихідних даних в залежності від значення параметру out_aligned.

```
proc sql noprint;
  select lengthn(put(max(&count_varname.), best. -L)) into
:decimals_num trimmed from &count_varname.;
quit;
```

На основі датасетів SUBJ_COUNT та EVNT_COUNT, що містять розраховані частоти в змінних числового типу, за допомогою data-кроку створюються датасети SUBJ_COUNT_C та EVNT_COUNT_C, в яких потрібні дані буде виведено у вигляді відформатованого тексту до символьних змінних. На базі отриманих значень частот та значень N (кількість суб'єктів популяції в конкретній стратифікаційній групі) розраховується відсоток суб'єктів, що мали подію як для групи категоризації (SOC) так і для кожного з PT. Величина N є знаменником, частота - чисельником, співвідношення помножується на 100%. Отриманий результат відсотків розміщується у дужках та додається після значення відповідної частоти. В залежності від налаштувань, що описані раніше, результуючий текстовий рядок вирівнюється – в потрібних місцях додаються пробіли.

На даному етапі також формується текст лейблу змінної з розрахунковим значенням для кожної з стратифікаційних груп. Він складається з назви групи (міститься у trt_var), слова «Subjects» або «Events» (в залежності від вхідного датасету), та значення N у дужках (відповідне до назви групи в trt_names значення макрозмінної trt_totals). Приклад результату: «Placebo Subjects (N=50)». Цей текст стає лейблом змінної на етапі транспонування датасету, для чого використовується оператор idlabel.

Код містить деякі перевірки, наприклад, на наявність даних та чи не дорівнює знаменник (значення N) нулю. Перед транспонуванням датасети SUBJ_COUNT та EVNT_COUNT сортуються за тими змінними, за яким потрібно виконати транспонування. В обох випадках вони вказуються через оператор by.

```
data &count_varname._c;
  set &count_varname.;
  length PCT VALUE TRT $200 DENOM 8;
  TRT = strip(&trt_var.) || ', '|| ifc("&count_varname." =
'SUBJ_COUNT', 'Subjects', 'Events') || ' (N=' ||
scan("&trt_totals.", findw("&trt_names.", strip(&trt_var.),
'|', 'E'), '|', 'R') || ')';
  if 0 then call missing(PCT, DENOM);

  %if &count_varname. = SUBJ_COUNT %then %do;
    DENOM = input(scan("&trt_totals.", &trt_varn., '|'),
best.);
    if DENOM  = &count_varname. then PCT = ('(100%)');
    else if DENOM ^= 0          then PCT = catt('(',
put(100*&count_varname./DENOM, %eval(&decimals. +
3).&decimals. &move_left.), '%)');
  %end;

  if DENOM ^= 0 then VALUE = trim(put(&count_varname.,
&decimals_num.. &move_left.)) || ' ' || PCT;
  else put "WARN" "ING: Macro &macro_name. %str(err)or:
Division by zero detected! (N=0 for &trt_var.=" &trt_var. ")";
run;

proc sort data = &count_varname._c out =
&count_varname._c_srt;
  by CAT &cat_var. &decod_varname. &term_varname.;
run;
```

```
proc transpose data = &count_varname._c_srt out =
&count_varname._c_tr (drop = _:) prefix =
%scan(&count_varname., 1, "_")_TRT;
  by CAT &cat_var. &decod_varname. &term_varname.;
  var VALUE;
  id &trt_varn.;
  idlabel TRT;
run;
```

Транспоновані датасети SUBJ_COUNT_c та EVNT_COUNT_c об'єднуються у один датасет percents оператором merge data-кроку за тим самим ключем, за яким відбувалося транспонування. Перед об'єднанням через оператор merge датасети необхідно обов'язково сортувати. У випадку коли розрахунок частот потрібно виконувати тільки для суб'єктів, тобто значення параметру do_calc_evt дорівнює false, сортування та об'єднання не відбувається. Створюваний датасет percents є копією транспонованого датасету SUBJ_COUNT_c.

```
%if %upcase(&do_calc_evt.) = TRUE %then %do;
  proc sort data = subj_count_c_tr;
    by CAT &cat_var. &decod_varname. &term_varname.;
  run;
  proc sort data = evnt_count_c_tr;
    by CAT &cat_var. &decod_varname. &term_varname.;
  run;

  data percents;
    merge subj_count_c_tr evnt_count_c_tr;
    by CAT &cat_var. &decod_varname. &term_varname.;
  run;
```

```
%end;
%else %do;
  data percents;
    set subj_count_c_tr;
  run;
%end;
```

Визначення назви змінної, за якою потрібно сортувати дані вихідного датасету, що відповідає стовпчику стратифікаційної групи у таблиці-звіті передається значенням параметру out_sort_by. Ця назва повинна відповідати одному із значень що присутні у змінній, назва якої вказана у параметрі trt_var. На даному етапі змінні датасету мають префікс SUBJ_TRT – оновлення назв змінних відповідно для параметру simple_varnames ще не відбулося. Тому достатньо перевірити який індекс (порядковий номер, що у макрозмінній trt_ords) імені групи у out_sort_by відповідає назві у макрозмінній trt_names.

Якщо відповідності не знайдено, що може бути, наприклад, у випадку коли значення у out_sort_by не відповідає жодному із значень, що передані через trt_var, то сортування буде відбуватися за змінною, що відповідає стратифікаційній групі «Total». Ім'я змінної для сортування записується до макрозмінної out_sort_varname.

```
%do i = 1 %to &trt_cnt.;
  %if "&out_sort_by." = "%scan(&trt_names., &i., |)" %then
%let out_sort_varname = SUBJ_TRT%scan(&trt_ords., &i., |);
%end;
%if &out_sort_varname. = %then %let out_sort_varname =
SUBJ_TRT%scan(&trt_ords., -1, |);
```

На базі датасету percents створюється змінна числового типу OUT_SRT_VAR, в яку записуються значення розрахованих частот. Для відсутніх

частот (місінгове значення) до змінної OUT_SRT_VAR буде записано нуль. Випадок відсутніх значень частот можливий коли сортування необхідно виконувати не за групою «Total». Тобто у суб'єктів одної стратифікаційної групи події були, а у суб'єктів другої групи - ні. Таким чином, частоти будуть присутніми тільки для першої групи та групи «Total».

```
data percents_ntot;
  set percents;
  length OUT_SRT_VAR 8;
  if missing(&out_sort_varname.) then OUT_SRT_VAR = 0;
  else OUT_SRT_VAR = input(scan(&out_sort_varname., 1),
best.);
  rename SUBJ_TRT%scan(&trt_ords., -1, |) =
SUBJ_%upcase(&total_col_name.);
  %if %upcase(&do_calc_evt.) = TRUE %then %do;
    rename EVNT_TRT%scan(&trt_ords., -1, |) =
EVNT_%upcase(&total_col_name.);
  %end;
run;
```

На даному етапі всі розрахунки виконано, потрібні форматовані текстові значення сформовані та записані до відповідних змінних. Наступним кроком є розміщення даних у необхідній послідовності. Для цього додаються змінні, які буде використано тільки для сортування вихідних даних.

За структурою, до таблиці виводяться всі категорії (SOC, ATC, тощо) до яких відносяться події (чи супутні препарати), що відбулися хоча б у одного з суб'єктів дослідження, незалежно від того, до якої стратифікаційної групи цей суб'єкт відноситься, та назви самих подій чи медикаментів. При чому спершу вказується назва категорії в першому стовпчику, а за нею нижче слідує перелік всіх подій, що до неї відносяться, а потім наступна категорія з подіями, що

відносяться вже до неї. В стовпчиках праворуч виводяться частоти та відсотки для кожної з наявних груп стратифікації (за потреби включаючи групу «Total»). Категорії розташовуються у порядку спадання кількості подій чи супутніх препаратів, які приймав суб’єкт. Слід відзначити, що в таблиці присутній рядок, що містить частоти без категоризації – тобто кількість подій, що відбулася, незалежно від належності до якоїсь категорії.

Як зазначалося раніше, для отримання потрібного порядку рядків в таблиці сортування відбувається в два етапи. Перший етап передбачає сортування самих категорій. На другому етапі аналогічним чином розташовуються події, що відносяться до категорії – перелік подій завжди прив'язаний до категорії. Код, який наведено нижче, відповідає за перший етап сортування. Датасет з даними спершу сортується за спаданням значень у змінній OUT_SRT_VAR всередині власного типу (змінна CAT). Якщо це перший рядок свого типу (наприклад для SOC тип буде дорівнювати «CAT_VAR»), то змінній CAT_ORD надається значення 2, та інкрементується для кожного наступного запису. Значення 1 зарезервоване для рядка без категоризації, що містить всі події (тип «ANY»). Аналогічним чином призначається значення для змінної tmp_PT_ORD, але нумерація починається з одиниці. Якщо рядок стосується не категорій, а подій, то змінна CAT_ORD очищається. Її значення повинно бути однаковим для всіх подій всередині категорії. Пізніше буде показано як на її основі розраховується інша, фінальна сортувальна змінна.

В цьому місці також відбувається генерація імен змінних відповідно до значення параметру simple_varnames. Фактично, якщо значення цього параметру не дорівнює true, відбувається перейменування змінних з вигляду SUBJ_TRT1 до вигляду PLACEBO_SUBJECTS (або TREATMENT_GROUP_A_SUBJECTS), де значення «PLACEBO» береться із отриманої раніше макрозмінної trt_vnames, що відповідає індексу «i» (лічильник циклу).

```
proc sort data = percents_ntot out = percents_ntot_srt;
  by CAT descending OUT_SRT_VAR &cat_var. &decod_varname.
&term_varname.;
run;
data cat_ord;
  length CAT_ORD tmp_PT_ORD 8;
  set percents_ntot_srt;
  by CAT descending OUT_SRT_VAR &cat_var. &decod_varname.
&term_varname.;
  if first.CAT then tmp_PT_ORD = 1;
  else tmp_PT_ORD + 1;
  if first.CAT then CAT_ORD = 2;
  else CAT_ORD    + 1;
  if CAT ^= 'CAT_VAR' then call missing(CAT_ORD);
  %if %upcase(&simple_varnames.) ^= TRUE %then %do;
    rename
      %do i = 1 %to &trt_c*nt. - 1;
        SUBJ_TRT&i. = %scan(&trt_vnames., &i., |)_SUBJECTS
        %if %upcase(&do_calc_evt.) = TRUE %then EVNT_TRT&i. =
%scan(&trt_vnames., &i., |)_EVENTS;
      %end;
      SUBJ_%upcase(&total_col_name.) =
&total_col_name._SUBJECTS
      %if %upcase(&do_calc_evt.) = TRUE %then
EVNT_%upcase(&total_col_name.) = &total_col_name._EVENTS;
  %end;
run;
```

Датасет отриманий на першому етапі створення сортувальних змінних сортується за назвами категорій (cat_var), а всередині них за типом (CAT) та

тимчасовою сортувальною змінною tmp_PT_ORD. Ця змінна отримана таким самим чином як і CAT_ORD, але на відміну від останньої, вона має значення для рядків з подіями (PT).

Додатково створюється ще одна тимчасова змінна tmp_ORD, значенням якої є порядок категорії. Значення в неї записується зі змінної CAT_ORD, якщо тип значень у рядку дорівнює CAT_VAR, а саме значення є незмінним при читанні наступних рядків вхідного датасету завдяки використанню оператора retain. Для всіх інших рядків, де тип даних не є CAT_VAR, відбувається зворотнє перенесення значень – зі змінної tmp_ORD до змінної CAT_ORD. Таким чином досягається однаковість значень змінної CAT_ORD для всіх однакових значень у змінній, що записана у параметрі cat_var. Далі призначається значення створеної на цьому data-кроці змінної PT_ORD – завдяки наявному сортуванню вхідного датасету, рядки розташовуються за спаданням величин частот подій, а отже PT_ORD прирівнюється до значення 1 для першого рядку з групи, де однакове значення cat_var.

Для типу записів CAT = «ANY» (рядок з частотами без використання категоризації) для змінної CAT_ORD встановлюється зарезервоване значення 1.

```
proc sort data = cat_ord out = cat_ord_srt;
  by &cat_var. CAT tmp_PT_ORD;
run;

data cat_pt_ord (drop = tmp:);
  format CAT &cat_var. &decod_varname. &term_varname. CAT_ORD
PT_ORD CAT_PT &out_vars_list.;
  length tmp_ORD PT_ORD 8 CAT_PT $200;
  set cat_ord_srt;
  by &cat_var. CAT tmp_PT_ORD;
  array VALUES &out_vars_list.;
  retain tmp_ORD;
```

```
  if CAT = 'CAT_VAR' then tmp_ORD = CAT_ORD;
  else CAT_ORD = tmp_ORD;
  if first.&cat_var. then PT_ORD = 1;
  else PT_ORD + 1;
  if CAT = 'ANY' then CAT_ORD = 1;

  select(CAT);
    when('ANY') CAT_PT = "&any_string_text.";
    when('CAT_VAR') CAT_PT = strip(&cat_var.);
    when('DECOD') CAT_PT = '  ' || strip(&decod_varname.);
    otherwise CAT_PT = '  ' || strip(&term_varname.);
  end;

  do over VALUES;
    if missing(VALUES) then VALUES = put(0, &decimals_num..
&move_left.);
  end;
run;
```

На цьому етапі також відбувається створення змінної CAT_PT, яка міститиме текстові значення для першого стовпчику таблиці – текст для рядка, де виводяться частоти без категоризації, назви категорії та подій (препаратів). Для відсутніх частот, коли не було суб'єктів з даною подією чи категорією для стратифікаційної групи, до змінної, що містить відформатовані значення частот (змінна VALUES) записується значення 0. Після нульового значення відсотки не виводяться, а саме значення вирівнюється горизонтально відповідно до вимог, таким самим чином, як це було зроблено для інших, наявних, значень.

Якщо параметром nc_put_last було встановлено вимогу перенести дані частот, що стосуються події з незакодованими категоріями (SOC), та подіями чи препаратами (PT) в низ таблиці, для відповідних рядків датасету оновлюється

значення сортувальної змінної CAT_ORD. Спочатку за допомогою функції max у процедурі proc sql знаходиться максимальне наявне значення змінної CAT_ORD та записується до макрозмінної cat_count. Потім всередині data-кроку для рядків, що відповідають незакодованим подіям значення змінної CAT_ORD інкрементується.

```
%if %upcase(&nc_put_last.) = TRUE %then %do;
  proc sql noprint;
    select max(CAT_ORD) into :cat_count trimmed from
cat_pt_ord;
  quit;

  data cat_pt_ord;
    set cat_pt_ord;
    if &cat_var. = "&nc_string_text." then CAT_ORD =
&cat_count. + 1;
  run;
%end;

proc sort data = cat_pt_ord;
  by CAT_ORD PT_ORD;
run;
```

Процес створення пустих вихідних датасетів та таблиці наведений нижче. Блок коду виконується у випадках, коли макрозмінна data_rec_num має значення 0, що означає відсутність вхідних даних для обробки. Раніше зазначалося, що виконання макросу переходить за допомогою макрооператора %goto до місця, позначеного міткою skip_calc. Вихідні датасети повинні мати необхідний перелік змінних із відповідними метаданими (тип, довжина, лейбл), але не мати жодного значення – кількість записів вихідних датасетів дорівнюватиме нулю. В такому

випадку ці датасети піддаються програмній обробці та система SAS буде здатна їх відкривати для відображення, та використовувати для обробки без виникнення помилок. Це, в першу чергу, необхідно для виклику процедури proc report, що безпосередньо створює звіт у вигляді таблиці, та дає змогу іншому програмісту (QC-програмісту) програмно обробляти отримані за використанням цього макросу датасети.

Порядок змінних вихідних датасетів визначається оператором format, а типи та довжини змінних – оператором length. Якщо змінні об'явлено, але жодного разу не використано, система SAS виведе попередження, а саме, повідомлення про відсутність ініціалізації змінних. Для пригнічення таких повідомлень всі змінні датасету обробляються функцією missing, яка спустошує значення змінних. SAS вважатиме це обробкою змінних, тому повідомлень не буде. Тут слід зазначити роботу частини «%sysfunc(tranwrd(&out_vars_list., %str( ), %str(,)))». Перелік змінних, що повинні використовуватися задля зберігання частот генерується макросом раніше та зберігається у макрозмінній out_vars_list. Але цей перелік має в якості роздільника пробіл, в той час, як для фукнкції missing є необхідним, щоб роздільником була кома. Тому на етапі макровиконання (тобто коли виконується формування коду для виконання даного data-кроку) функцією tranwrd пробіли у макрозмінній out_vars_list заміняються на кому з пробілом, та підставляються в місце, де викликано %sysfunc. Макрофункція %sysfunc забезпечує виконання функції data-кроку tranwrd саме на етапі макрообробки, а не під час виконання data-кроку. Таким чином, коли настане час виконання самого data-кроку, рядок «call missing(PAGE_ORD, CAT_ORD, TOGGLE, PT_ORD, CAT_PT, %sysfunc(tranwrd(&out_vars_list., %str( ), %str(,))));»
виглядатиме як:
«call missing(PAGE_ORD, CAT_ORD, TOGGLE, PT_ORD, CAT_PT, VAR1, VAR2, VAR3);»,
де VAR1 – VAR3 змінні, що буде підставлено (назви змінних спрощено для простоти пояснення).

```
%skip_calc:;

%if &data_rec_num. = 0 %then %do;
  data report_dsin;
    format PAGE_ORD CAT_ORD TOGGLE PT_ORD CAT_PT
&out_vars_list.;
    length TOGGLE CAT_PT &out_vars_list. $200 PAGE_ORD CAT_ORD
PT_ORD 8;
    call missing(PAGE_ORD, CAT_ORD, TOGGLE, PT_ORD, CAT_PT,
%sysfunc(tranwrd(&out_vars_list., %str( ), %str(, ))));
  run;
%end;

data &dsout. (keep = CAT_PT &out_vars_list.);
  set report_dsin;
  if &data_rec_num. ^= 0 then output;
run;
```

В залежності від умов виконання макросу (з якими значеннями параметрів його було викликано) змінюється кількість датасетів, яку він створює. Тому, у випадку, коли через параметр do_cleanup замовлена очистка середовища від «зайвих» датасетів, важливо знати, які датасети створювалися, а які ні. Якщо спробувати знищити датасет якого не існує, виникне помилка. Тому до простого виклику drop table процедури proc sql додається обробка умовними макрооператоррми (%if-%then).

```
%if %upcase(&do_cleanup.) = TRUE %then %do;
  proc sql noprint;
```

```
    drop table adsl_data, input_data, trtgrp_names,
trtgrp_names_tr, trtgrp_vnames, sl_input_data, fmt_ds;
    %if &data_rec_num. ^= 0 %then %do;
      drop table subj_count, subj_count_c, subj_count_c_srt,
subj_count_c_tr, percents, percents_ntot, percents_ntot_srt,
cat_ord, cat_pt_ord, cat_ord_srt;
      %if %upcase(&do_calc_evt.) = TRUE %then %do;
        drop table evnt_count, evnt_count_c, evnt_count_c_srt,
evnt_count_c_tr;
      %end;
    %end;
  quit;
%end;
```

Вивід інформації щодо обробки даних макросом до логу програми відбувається на етапі макробробки шляхом виклику оператору %put. Цей оператор виводить текст, що слідує за його викликом, та в змозі читати значення змінних та макрозмінних для виведення значень, що знаходяться всередині них. Інформацію буде представлено у вигляді таблиці, в якій будуть присутні наступні дані: назви стратифікаційних груп, кількість суб'єктів популяції, що до них відносяться, порядок, в якому ці групи буде представлено у вихідних датасетах, та таблиці, індикатор (знак «+») навпроти групи, за якою відбувалося сортування вихідних даних. Ця таблиця виводиться до логу програми тільки коли макрос викликано із значенням true параметру do_out_info, вигляд таблиці наведено на рис. 10.

```
NOTE: SOC_PT Treatment Groups Info:
      Number of treatment groups: 3
      Treatment Group | Subjects (N=) | Order | Sorted by
      Stratum A       | 24            | 1     |
      Stratum B       | 22            | 2     |
      Total           | 46            | 3     | +
```

Рисунок 10 – Приклад інформаційної таблиці у лозі програми

```
%if %upcase(&do_out_info.) = TRUE %then %do;
  %let length_max = 15;
  %do i = 1 %to &trt_cnt.;
    %if %eval(&length_max. < %length(%scan(&trt_names., &i.,
|))) %then %let length_max = %length(%scan(&trt_names., &i.,
|));
  %end;
  %let out_log_text = %sysfunc(putc(Treatment Group,
&length_max..)) | Subjects (N=) | Order | Sorted by;
  %put;
  %put NOTE- ----------------------------------------------
--;
  %put NOTE: &macro_name. Treatment Groups Info:;
  %put NOTE- Number of treatment groups: &trt_cnt.;
  %put NOTE- &out_log_text.;
  %do i = 1 %to &trt_cnt.;
    %let out_log_text = %sysfunc(putc(%scan(&trt_names., &i.,
|), &length_max..)) | %sysfunc(putc(%scan(&trt_totals., &i.,
|), 13.)) | %sysfunc(putc(%scan(&trt_ords., &i., |), 5.)) |;
    %if &out_sort_varname. = SUBJ_TRT%scan(&trt_ords., &i., |)
%then %let out_log_text = &out_log_text. %str(+);
    %put NOTE- &out_log_text.;
  %end;
  %put NOTE- ----------------------------------------------
--;
  %put;
%end;
```

Код для створення таблиці звіту з результатами обчислень наведено нижче. Це стандартний виклик SAS-процедури proc report, де для досягнення потрібного вигляду (та використання потрібних змінних, які генеруються в процесі виконання макросу) використовуються макропідстановки. Серед таких підстановок стилі – текст форматується відповідно до сформованого на початку виконання макроса значення макрозмінної out_style; змінні – назви змінних та їхня кількість формуються в процесі обробки макросом даних та зберігається у макрозмінній out_vars_list; текст заголовків стовпчиків, наприклад, з макрозмінної (параметру) cat_header_text. Тип та розташування вихідного файлу з таблицею задається також в цьому місці, через оператор ods вказується тип (параметр rtf_pdf), а через опцію file = оператору ods шлях до файлу та його ім'я (параметр макросу out_file).

```
%if %upcase(&do_out_report.) = TRUE %then %do;
  ods listing close;
  ods &rtf_pdf. file = "&out_file.";
  ods escapechar = '^';

  %let out_trt_col_width =
%sysfunc(round(%sysevalf(%sysfunc(max(&decimals_num. + 8,
%length(Subjects)))*2.86/46 + %sysevalf(2*&hor_margin. +
0.02)), 0.01));
  proc report data = report_dsin split = '*' missing headskip
nowd style(report) = {width = 100%} style(column) =
{&out_style.} style(header) = {&out_style. just = center};
    column PAGE_ORD CAT_ORD TOGGLE PT_ORD CAT_PT
      %do i = 1 %to &trt_cnt. - %eval(%upcase(&do_calc_total.)
^= TRUE);
        ("%scan(&trt_names., &i., |)^n(N=%scan(&trt_totals.,
&i., |))"
```

```
        %if %upcase(&do_calc_evt.) = TRUE %then
%scan(&out_vars_list., 2*&i. - 1, %str( ))
%scan(&out_vars_list., 2*&i., %str( )));
        %else %scan(&out_vars_list.,   &i.,     %str( )));
      %end;
    ;
    define PAGE_ORD /order = data noprint;
    define CAT_ORD  /order = data noprint;
    define TOGGLE   /group order = data noprint;
    define PT_ORD   /order = data noprint;
    define CAT_PT   /group order = data "&cat_header_text."
style(column) = {width = 4.0in} style(header) = {just = left};

    %do i = 1 %to &trt_cnt. - %eval(%upcase(&do_calc_total.)
^= TRUE);
      %if %upcase(&do_calc_evt.) = TRUE %then %do;
        define %scan(&out_vars_list., 2*&i. - 1, %str( ))
/display order = data "Subjects" style(column) = {width =
&out_trt_col_width.in} style(header) = {just = left};
        define %scan(&out_vars_list., 2*&i., %str( )) /display
order = data "Events" style(column) = {width =
&out_trt_col_width.in} style(header) = {just = left};
      %end;
      %else %do;
        define %scan(&out_vars_list., &i., %str( )) /display
order = data "Subjects" style(column) = {width =
&out_trt_col_width.in} style(header) = {just = left};
      %end;
    %end;
```

```
    break after PAGE_ORD /page;
    compute before TOGGLE   /style = {&out_style.};
      if TOGGLE = CAT_PT then TOGGLE = '';
      line TOGGLE $200.;
    endcomp;
    %if &data_rec_num. = 0 %then %do;
      compute before CAT_PT /style = {&out_style. just =
center};
        line "&nodata_string_text.";
      endcomp;
    %end;
  run;

  ods &rtf_pdf. close;
  ods listing;
%end;
```

Кінцевим результатом роботи SOC_PT макросу є створення таблиці-звіту, перша сторінка якого для прикладу наведена на рис. 11. Звіт містить чотири стовпчики, три з яких містять дані частот суб'єктів клінічного дослідження що мали побічний ефект, назва якого наведена у першому стовпчику таблиці.

Table 1.1
Summary of Adverse Events by System Organ Class and Preferred Term

Population: Safety

System Organ Class Preferred Term	Stratum A N=24 n (%)	Stratum B N=22 n (%)	Total N=46 n (%)
Number of subjects with at Least One Event	24 (100.0%)	22 (100.0%)	46 (100.0%)
Gastrointestinal disorders	22 (91.7%)	18 (81.8%)	40 (87.0%)
Diarrhoea	18 (75.0%)	18 (81.8%)	36 (78.3%)
Nausea	15 (62.5%)	9 (40.9%)	24 (52.2%)
Vomiting	6 (25.0%)	10 (45.5%)	16 (34.8%)
Constipation	5 (20.8%)	4 (18.2%)	9 (19.6%)
Abdominal pain	2 (8.3%)	4 (18.2%)	6 (13.0%)
General disorders and administration site conditions	19 (79.2%)	18 (81.8%)	37 (80.4%)
Pyrexia	12 (50.0%)	11 (50.0%)	23 (50.0%)
Fatigue	7 (29.2%)	9 (40.9%)	16 (34.8%)
Pain	3 (12.5%)	4 (18.2%)	7 (15.2%)
Asthenia	2 (8.3%)	4 (18.2%)	6 (13.0%)
Oedema peripheral	3 (12.5%)	1 (4.5%)	4 (8.7%)
Non-cardiac chest pain	1 (4.2%)	2 (9.1%)	3 (6.5%)
Malaise	0	2 (9.1%)	2 (4.3%)
Impaired healing	1 (4.2%)	0	1 (2.2%)
Influenza like illness	0	1 (4.5%)	1 (2.2%)
Vessel puncture site haematoma	1 (4.2%)	0	1 (2.2%)

Note: Percentages are based on the number of subjects in safety population.
Subjects are counted once within each system organ class and each preferred term.
All investigator adverse event terms were coded using MedDRA dictionary version 26.0.

Рисунок 11 – Таблиця звіт, що згенерована в результаті виконання макросу SOC_PT

2.6 Тестування макросу

Випробування працездатності, безпомилковості роботи та відповідності заявленим функціональним спроможностям макросу SOC_PT відбувалося в декілька етапів.

За використанням розробленого макросу було створено 100 таблиць-звітів та відповідних ним датасетів. 50 з яких були по побічним ефектам (10 з цих 50 запускалися за використанням або пустих вхідних датасетів, або з такими

фільтрами, що призводили до відсутності даних для аналізу), 30 – по супутнім медикаментам, 10 – по медичним процедурам та 10 – по записам анамнезу. На боці валідації виконувалися аналогічні розрахунки, що зроблені за використанням поєднання вузькоспеціалізованих макросів та відкритого (без надбудов у вигляді макросів тощо) коду.

Датасети, згенеровані макросом та програмістом-валідатором, порівнювалися між собою з метою виявлення помилок розрахунків у розробленому макросі, а також за для виявлення невідповідностей щодо вимог форматування розрахункових даних.

Таблиці-звіти у RTF-форматі перевірялися вручну (без застосування програмних засобів) також програмістом валідатором на предмет відповідності між даними, що присутні у згенерованих датасетах, та даними, що представлені в таблиці-звіті. Також перевірялися вірність заповнення шапок колонок таблиці та вірність форматування вихідних даних з урахуванням назв стратифікаційних груп та кількості суб'єктів використовуваної вибірки.

Додатково до вищезазначеного виконано перевірку логу, що генерує SAS-програма, яка викликає код макросу, на предмет наявності помилок, попереджень та небажаних повідомлень, таких як неявна конвертація даних (з числового до символьного та навпаки), відсутність ініціалізації змінних чи операції над місінговими величинами. Перевірка виконувалася спеціалізованим макросом для перевірки логів SAS-програм, який є валідованим, тобто є беззаперечно надійним та перевіреним інструментом.

Результати перевірки не виявили відмінностей між результатами, що отримані макросом та програмістом-валідатором, як з точки зору розрахованих величин, так і з точки зору форматування. Перевірка логу виконання SAS-програми з викликом макросу також не виявила присутності помилок чи неочікуваної поведінки макросу.

ВИСНОВКИ

В результаті виконання роботи розроблено макропрограму, яка виконує обробку SDTM- й ADaM-датасетів клінічних досліджень і генерує TLF-звіти та звіти-датасети.

Гнучкість виконання основного коду програми досягається завдяки використанню SAS Macro Facility, який дає змогу змінювати код виконуваної програми в залежності від додаткових умов.

Розроблений макрос SOC_PT здатен значною мірою модифікувати зовнішній вид представлення даних, як вихідних датасетів, так і таблиці-звіту, тим самим надаючи змогу використовувати його для створення звітів на базі побічних ефектів, що спостерігалися у суб'єктів клінічних досліджень, прийманих ними супутніх ліків, проходження медичних процедур, а також наявних записів у анамнезі.

Широкий асортимент налаштувань вхідних параметрів макросу зумовлює можливість адаптації отримуваних результатів згідно вимогам будь-якого замовника – фармкомпанії чи CRO.

Швидкість виконання макросу забезпечує високу ефективність роботи клінічних програмістів, незалежно від безпосередньо виконуваної ними частини згідно до концепції подвійного програмування – результати роботи макросу задовільняють вимогам обох сторін, як первинних програмістів, так і програмістів-валідаторів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. URL: https://www.who.int/health-topics/clinical-trials (дата звернення 26.02.2024)
2. ICH Harmonised Guideline – Integrated Addendum to ICH E6 (R1): Guideline For Good Clinical Practice E6(R2). Step 4 version, 2016.
3. Clinical Data Interchange Standards Consortium, Inc. *Study Data Tabulation Model Implementation Guide: Human Clinical Trials*. Version 3.4, 2021, 461 p.
4. URL: https://www.cdisc.org/standards (дата звернення 26.02.2024)
5. URL: https://www.altexsoft.com/blog/cdisc-standards. (дата звернення 27.02.2024)
6. URL: https://www.cdisc.org/standards/foundational/protocol (дата звернення 27.02.2024)
7. CDASH Data Definitions and CRF Examples. URL: https://www.cdisc.org/standards/foundational/cdash (дата звернення 28.03.2024)
8. CDISC CDASH Team, *Clinical Data Acquisition Standards Harmonization (CDASH)*. Version 1.1, 2011, 203 p.
9. Clinical Data Interchange Standards Consortium, Inc. *Analysis Data Model (ADaM)*. Version 2.1, 2009, 41 p.
10. URL: https://www.cdisc.org/standards/terminology/controlled-terminology (дата звернення 28.03.2024)
11. URL: https://www.pinnacle21.com/news/opencdisc-validator-15-available-now (дата звернення 28.03.2024)
12. CDISC Submission Data Standards Team, Study Data Tabulation Model. Version 2.0, 2021, 74 p.
13. Clinical Data Interchange Standards Consortium, Inc. Analysis Data Model Implementation Guide. Version 1.2, 2019, 110 p.

14. A. Sean-Paul. Bespoke outputs, just like everyone else's: using the SAS® Macro Language to create template programs for standard presentations of your client's clinical trial data. // *PharmaSUG*, Paper 192, 2019, 9 p.
15. D. Li, S. Li, S. Sproule. A SAS Macro for Creating Adverse Event Analysis Dataset. Paper CC06 // *PharmaSUG*, Paper CC06, 10 p.
16. Y. Zhiping, P. Dizal. Interlocking and Mutually Reinforcing System - From Shells to TLGs Automatically Generation. // *PharmaSUG*, Paper AD-134, China, 2022, 12 p.
17. Shostak. SAS® Programming in the Pharmaceutical Industry. Cary, NC: SAS Institute Inc., 2005, 355 p.
18. Ph. Spector. An Introduction to the SAS System. Berkeley, University of California, 2013, 168 p.
19. R. Cody. Learning SAS by Example: A Programmer's Guide. Cary, NC: SAS Institute Inc., 2007, 626 p.

ДОДАТОК Код макросу для створення TLF-звітів

```
%macro SOC_PT(dsin_SL = ADSL, dsin_DATA = ADAE, domain = AE,
dsout =, trt_var = TRT01A, trt_varn =, cat_var =, pop_var =,
filter =, do_calc_evt = false, do_calc_total = true,
total_col_name = Total, out_sort_by = total, decimals = 1,
out_aligned = false, any_string_text = Subjects with Any
Adverse Event, nc_put_last = false, nc_string_text = Not
Coded, do_cleanup = true, do_out_info = false, do_out_report =
true, lines_ppage = 20, nodata_string_text = *****No Data
Available for Processing******, cat_pt_linelen = 60,
cat_header_text = System Organ Class /Preferred Term, out_file
=, rtf_pdf = RTF, simple_varnames = false);

  %global trt_cnt trt_names trt_totals trt_ords;
  %local  macro_name tot_subjs count_varname count_mode
decimals_num move_left out_sort_varname out_vars_list
cat_count length_max out_log_text simple_varnames data_rec_num
out_style hor_margin out_trt_col_width decod_varname
term_varname;

  %let macro_name = SOC_PT;
  %let hor_margin = 0.00;
  %let out_style = just = left asis = on font_face = Arial
font_size = 10 marginleft = &hor_margin.in marginright =
&hor_margin.in;

  %let simple_varnames = FALSE;
```

```
  %if %upcase(&out_aligned.) ^= TRUE %then %let move_left = -
L;
  %else %let move_left =;
  %if &dsout. = %then %let dsout = soc_pt_out;
  %if &decimals. = %then %let decimals = 1;
  %if &lines_ppage. = %then %let lines_ppage = 40;
  %if "&out_sort_by." = "" or "%upcase(&out_sort_by.)" =
"OVERALL" %then %let out_sort_by = TOTAL;
  %if "&total_col_name." = "" %then %let total_col_name =
Total;
  %if "&any_string_text." = "" %then %let any_string_text =
Subjects with Any &domain.;
  %if "&nc_string_text." = "" %then %let nc_string_text = Not
Coded;
  %if "&nodata_string_text." = "" %then %let
nodata_string_text = *****No Data Available for
Processing******;
  %if "&cat_header_text." = "" %then %let cat_header_text =
System Organ Class /Preferred Term;
  %if "&out_file." = "" %then %let out_file = C:\soc_pt.rtf;
  %if "&rtf_pdf." = "" %then %let rtf_pdf = RTF;

  %if %upcase(&domain.) = AE %then %do;
    %let decod_varname = AEDECOD;
    %let term_varname = AETERM;
  %end;
  %else %if %upcase(&domain.) = CM %then %do;
    %let decod_varname = CMDECOD;
    %let term_varname = CMTRT;
```

```
  %end;
  %else %if %upcase(&domain.) = MH %then %do;
    %let decod_varname = MHDECOD;
    %let term_varname = MHTERM;
  %end;
  %else %if %upcase(&domain.) = PR %then %do;
    %let decod_varname = PRDECOD;
    %let term_varname = PRTRT;
  %end;
  %else %do;
    %put %str(ERR)OR: Macro &macro_name. %str(err)or: The only
acceptable values for the domain parameter are AE, CM, MH or
PR. The macro will exit now.;
    %return;
  %end;

  %if &dsin_SL. = or &dsin_DATA. = or &domain. = or &trt_var.
= or &cat_var. = %then %do;
    %put %str(ERR)OR: Macro &macro_name. %str(err)or: All core
parameters (dsin_SL, dsin_DATA, domain, trt_var, cat_var) must
be provided. The macro will exit now.;
    %return;
  %end;

  proc sort data = &dsin_SL. out = adsl_data;
    by STUDYID USUBJID;
    where %if &pop_var. ^= %then &pop_var. = 'Y';;
  run;
```

```
  proc sql noprint;
    create table fmt_ds as
      select distinct 'ifmt_trtn' as FMTNAME,
             upcase(&trt_var.) as START,
             'i' as TYPE,
             'U ' as HLO
             %if &trt_varn. ^= %then %str(, &trt_varn. as
LABEL);
      from adsl_data
      %if &trt_varn. = %then %str(order by START);
      %else %str(order by LABEL);
    ;
  quit;

  data fmt_ds;
    length LABEL 8;
    set fmt_ds end = EOF;
    %if &trt_varn. = %then %str(LABEL + 1);;
    output;
    if EOF then do;
      START = upcase("&total_col_name.");
      LABEL = LABEL + 1;
      output;
    end;
  run;

  proc sort data = fmt_ds;
    by START;
  run;
```

```
  proc format cntlin = fmt_ds;
run;

%if &trt_varn. = %then %do;
  %let trt_varn = &trt_var._N;
  data adsl_data;
    length &trt_varn. 8;
    set adsl_data;
    &trt_varn. = input(&trt_var., ifmt_trtn.);
  run;
%end;

proc sort data = adsl_data;
  by STUDYID USUBJID;
run;

proc sort data = &dsin_DATA. out = input_data;
  by STUDYID USUBJID;
run;

data sl_input_data;
  merge adsl_data (in = dsin_SL)
        input_data (in = dsin_DATA);
  by STUDYID USUBJID;
  if dsin_SL and dsin_DATA;
run;

proc sql noprint;
```

```
    select n(distinct &trt_var.),
           n(distinct USUBJID)
           into :trt_cnt trimmed,
                :tot_subjs trimmed
      from adsl_data
    ;
    select distinct &trt_var.,
                    n(distinct USUBJID),
                    &trt_varn.
             into :trt_names separated by '|',
                  :trt_totals separated by '|',
                  :trt_ords separated by '|'
      from adsl_data
      group by &trt_var.
      order by &trt_varn.
    ;
  quit;

  proc sort data = adsl_data out = trtgrp_names (keep =
&trt_var. &trt_varn.) nodupkey;
    by &trt_varn.;
  run;

  proc transpose data = trtgrp_names out = trtgrp_names_tr
(drop = _:);
    var &trt_varn.;
    id &trt_var.;
  run;
```

```
  proc transpose data = trtgrp_names_tr out = trtgrp_vnames
(rename = (COL1 = &trt_varn.)) NAME = &trt_var._VNAME;
    var _all_;
  run;

  proc sql noprint;
    select distinct &trt_var._VNAME,
                    &trt_varn.
      into :trt_vnames separated by '|',
           :trt_vords separated by '|'
      from trtgrp_vnames
      order by &trt_varn.
    ;
  quit;

  %let trt_cnt = %eval(&trt_cnt. + 1);
  %let trt_names = &trt_names.|&total_col_name.;
  %let trt_vnames = &trt_vnames.|&total_col_name.;
  %let trt_totals = &trt_totals.|&tot_subjs.;
  %let trt_ords = &trt_ords.|%sysfunc(inputn(&total_col_name.,
ifmt_trtn.));
  %let trt_vords =
&trt_vords.|%sysfunc(inputn(&total_col_name., ifmt_trtn.));

  %if "&trt_ords." ^= "&trt_vords." %then %put %str(WARN)ING:
Macro &macro_name. unhandled exception: output variables may
not correctly represent treatment groups.;

  data sl_input_data;
```

```
    set sl_input_data;
    output;
    &trt_var. = "&total_col_name.";
    &trt_varn. = input(&trt_var., ifmt_trtn.);
    output;
  run;

  data sl_input_data;
    set sl_input_data;
    where
      %if &pop_var. ^= %then &pop_var. = 'Y';
      %if &pop_var. ^= and &filter. ^= %then %str( and );
      %if &filter. ^= %then &filter.;
    ;
  run;

  %if %upcase(&simple_varnames.) ^= TRUE %then %do;
    %do k = 1 %to &trt_cnt. - 1;
      %let out_vars_list = &out_vars_list. %scan(&trt_vnames.,
&k., |)_SUBJECTS;
      %if %upcase(&do_calc_evt.) = TRUE %then %let
out_vars_list = &out_vars_list. %scan(&trt_vnames., &k.,
|)_EVENTS;
    %end;
    %if %upcase(&do_calc_total.) = TRUE %then %do;
       %let out_vars_list = &out_vars_list.
&total_col_name._SUBJECTS;
       %if %upcase(&do_calc_evt.) = TRUE %then %let
out_vars_list = &out_vars_list. &total_col_name._EVENTS;
```

```
    %end;
  %end;
  %else %do;
    %do k = 1 %to &trt_cnt. - 1;
      %let out_vars_list = &out_vars_list. SUBJ_TRT&k.;
      %if %upcase(&do_calc_evt.) = TRUE %then %let
out_vars_list = &out_vars_list. EVNT_TRT&k.;
    %end;
    %if %upcase(&do_calc_total.) = TRUE %then %do;
      %let out_vars_list = &out_vars_list.
SUBJ_%upcase(&total_col_name.);
      %if %upcase(&do_calc_evt.) = TRUE %then %let
out_vars_list = &out_vars_list.
EVNT_%upcase(&total_col_name.);
    %end;
  %end;

  data _null_;
    call symputx("data_rec_num", OBS_CNT);
    if 0 then set sl_input_data nobs = OBS_CNT;
  run;

  %if &data_rec_num. = 0 %then %goto skip_calc;

  %do i = 1 %to %eval(1 + (%upcase(&do_calc_evt.) = TRUE));
    %if &i. = 1 %then %do;
      %let count_varname = SUBJ_COUNT;
      %let count_mode  = %str(distinct);
    %end;
```

```
    %else %do;
      %let count_varname = EVNT_COUNT;
      %let count_mode  = %str();
    %end;
    proc sql noprint;
      create table &count_varname. as
          select n(&count_mode. USUBJID) as &count_varname.,
&trt_var., &trt_varn., 'ANY' as CAT
            from sl_input_data
            group by &trt_var., &trt_varn.
          outer union corr
            select n(&count_mode. USUBJID) as &count_varname.,
&trt_var., &trt_varn., &cat_var., 'CAT_VAR' as CAT
            from sl_input_data
            where ^missing(&cat_var.)
            group by &trt_var., &trt_varn., &cat_var.
          outer union corr
            select n(&count_mode. USUBJID) as &count_varname.,
&trt_var., &trt_varn., "&nc_string_text." as &cat_var.,
'CAT_VAR' as CAT
            from sl_input_data
            where missing(&cat_var.)
            group by &trt_var., &trt_varn., &cat_var.
          outer union corr
            select n(&count_mode. USUBJID) as &count_varname.,
&trt_var., &trt_varn., &cat_var., &decod_varname., 'DECOD' as
CAT
            from sl_input_data
            where ^missing(&cat_var.)
```

```
            group by &trt_var., &trt_varn., &cat_var.,
&decod_varname.
          outer union corr
            select n(&count_mode. USUBJID) as &count_varname.,
&trt_var., &trt_varn., "&nc_string_text." as &cat_var.,
&term_varname., 'TERM' as CAT
            from sl_input_data
            where missing(&cat_var.)
            group by &trt_var., &trt_varn., &cat_var.,
&term_varname.
          order by &trt_varn., &cat_var., CAT,
&decod_varname., &term_varname.
      ;
    quit;

    proc sql noprint;
      select lengthn(put(max(&count_varname.), best. -L)) into
:decimals_num trimmed from &count_varname.;
    quit;

    data &count_varname._c;
      set &count_varname.;
      length PCT VALUE TRT $200 DENOM 8;
      TRT = strip(&trt_var.) || ', '|| ifc("&count_varname." =
'SUBJ_COUNT', 'Subjects', 'Events') || ' (N=' ||
scan("&trt_totals.", findw("&trt_names.", strip(&trt_var.),
'|', 'E'), '|', 'R') || ')';
      if 0 then call missing(PCT, DENOM);
```

```
      %if &count_varname. = SUBJ_COUNT %then %do;
        DENOM = input(scan("&trt_totals.", &trt_varn., '|'),
best.);
        if DENOM = &count_varname. then PCT = ('(100%)');
        else if DENOM ^= 0         then PCT = catt('(',
put(100*&count_varname./DENOM, %eval(&decimals. +
3).&decimals. &move_left.), '%)');
      %end;

      if DENOM ^= 0 then VALUE = trim(put(&count_varname.,
&decimals_num.. &move_left.)) || ' ' || PCT;
      else put "WARN" "ING: Macro &macro_name. %str(err)or:
Division by zero detected! (N=0 for &trt_var.=" &trt_var. ")";
    run;

    proc sort data = &count_varname._c out =
&count_varname._c_srt;
      by CAT &cat_var. &decod_varname. &term_varname.;
    run;

    proc transpose data = &count_varname._c_srt out =
&count_varname._c_tr (drop = _:) prefix =
%scan(&count_varname., 1, "_")_TRT;
      by CAT &cat_var. &decod_varname. &term_varname.;
      var VALUE;
      id &trt_varn.;
      idlabel TRT;
    run;
  %end;
```

```
  %if %upcase(&do_calc_evt.) = TRUE %then %do;
    proc sort data = subj_count_c_tr;
      by CAT &cat_var. &decod_varname. &term_varname.;
    run;
    proc sort data = evnt_count_c_tr;
      by CAT &cat_var. &decod_varname. &term_varname.;
    run;

    data percents;
      merge subj_count_c_tr evnt_count_c_tr;
      by CAT &cat_var. &decod_varname. &term_varname.;
    run;
  %end;
  %else %do;
    data percents;
      set subj_count_c_tr;
    run;
  %end;

  %do i = 1 %to &trt_cnt.;
    %if "&out_sort_by." = "%scan(&trt_names., &i., |)" %then
%let out_sort_varname = SUBJ_TRT%scan(&trt_ords., &i., |);
  %end;
  %if &out_sort_varname. = %then %let out_sort_varname =
SUBJ_TRT%scan(&trt_ords., -1, |);

  data percents_ntot;
    set percents;
```

```
    length OUT_SRT_VAR 8;
    if missing(&out_sort_varname.) then OUT_SRT_VAR = 0;
    else OUT_SRT_VAR = input(scan(&out_sort_varname., 1),
best.);
    rename SUBJ_TRT%scan(&trt_ords., -1, |) =
SUBJ_%upcase(&total_col_name.);
    %if %upcase(&do_calc_evt.) = TRUE %then %do;
      rename EVNT_TRT%scan(&trt_ords., -1, |) =
EVNT_%upcase(&total_col_name.);
    %end;
  run;

  proc sort data = percents_ntot out = percents_ntot_srt;
    by CAT descending OUT_SRT_VAR &cat_var. &decod_varname.
&term_varname.;
  run;

  data cat_ord;
    length CAT_ORD tmp_PT_ORD 8;
    set percents_ntot_srt;
    by CAT descending OUT_SRT_VAR &cat_var. &decod_varname.
&term_varname.;
    if first.CAT then tmp_PT_ORD = 1;
                 else tmp_PT_ORD + 1;
    if first.CAT then CAT_ORD    = 2;
                 else CAT_ORD    + 1;
    if CAT ^= 'CAT_VAR' then call missing(CAT_ORD);
    %if %upcase(&simple_varnames.) ^= TRUE %then %do;
      rename
```

```
        %do i = 1 %to &trt_cnt. - 1;
          SUBJ_TRT&i. = %scan(&trt_vnames., &i., |)_SUBJECTS
          %if %upcase(&do_calc_evt.) = TRUE %then EVNT_TRT&i.
= %scan(&trt_vnames., &i., |)_EVENTS;
        %end;
        SUBJ_%upcase(&total_col_name.) =
&total_col_name._SUBJECTS
        %if %upcase(&do_calc_evt.) = TRUE %then
EVNT_%upcase(&total_col_name.) = &total_col_name._EVENTS;
     ;
    %end;
  run;

  proc sort data = cat_ord out = cat_ord_srt;
    by &cat_var. CAT tmp_PT_ORD;
  run;

  data cat_pt_ord (drop = tmp:);
    format CAT &cat_var. &decod_varname. &term_varname.
CAT_ORD PT_ORD CAT_PT &out_vars_list.;
    length tmp_ORD PT_ORD 8 CAT_PT $200;
    set cat_ord_srt;
    by &cat_var. CAT tmp_PT_ORD;
    array VALUES &out_vars_list.;
    retain tmp_ORD;
    if CAT = 'CAT_VAR' then tmp_ORD = CAT_ORD;
    else CAT_ORD = tmp_ORD;
    if first.&cat_var. then PT_ORD  = 1;
    else PT_ORD  + 1;
```

```
    if CAT = 'ANY' then CAT_ORD = 1;

    select(CAT);
      when('ANY') CAT_PT = "&any_string_text.";
      when('CAT_VAR') CAT_PT = strip(&cat_var.);
      when('DECOD') CAT_PT = '  ' || strip(&decod_varname.);
      otherwise CAT_PT = '  ' || strip(&term_varname.);
    end;

    do over VALUES;
      if missing(VALUES) then VALUES = put(0, &decimals_num..
&move_left.);
    end;
  run;

  %if %upcase(&nc_put_last.) = TRUE %then %do;
    proc sql noprint;
      select max(CAT_ORD) into :cat_count trimmed from
cat_pt_ord;
    quit;

    data cat_pt_ord;
      set cat_pt_ord;
      if &cat_var. = "&nc_string_text." then CAT_ORD =
&cat_count. + 1;
    run;
  %end;

  proc sort data = cat_pt_ord;
```

```
    by CAT_ORD PT_ORD;
  run;

  data report_dsin;
    format PAGE_ORD CAT_ORD TOGGLE PT_ORD CAT_PT
&out_vars_list.;
    length PAGE_ORD LINE_NUM _CUR_LINE_LEN _WORDS_IN_LINE 8
TOGGLE _CURRENT_WORD $200;
    set cat_pt_ord;
    by CAT_ORD PT_ORD;
    retain PAGE_ORD 1 LINE_NUM TOGGLE;
    LINE_NUM + 1;

    if first.CAT_ORD then do;
      LINE_NUM + 1;
      if TOGGLE = ' ' then TOGGLE = '  ';
                      else TOGGLE = ' ';
    end;

    %if &cat_pt_linelen. ^= %then %do;
      _CUR_LINE_LEN = 0;
      _WORDS_IN_LINE = countw(CAT_PT, ' ');
      do i = 1 to _WORDS_IN_LINE;
        _CURRENT_WORD = scan(CAT_PT, i, ' ');
        _CUR_LINE_LEN = _CUR_LINE_LEN + lengthn(_CURRENT_WORD)
+ (i ^= _WORDS_IN_LINE);
        if _CUR_LINE_LEN > &cat_pt_linelen. then do;
          LINE_NUM + 1;
          _CUR_LINE_LEN = lengthn(_CURRENT_WORD);
```

```
        end;
      end;
    %end;


    if LINE_NUM > &lines_ppage. or (first.CAT_ORD and LINE_NUM
= &lines_ppage.) then do;
      PAGE_ORD + 1;
      LINE_NUM = 2;
      if PAGE_ORD > 1 and CAT_PT ^= &cat_var. and
^first.CAT_ORD then do;
        TOGGLE = '^n' || &cat_var.;
        LINE_NUM + 1;
      end;
    end;
  run;


  proc sort data = report_dsin (keep = PAGE_ORD CAT_ORD TOGGLE
PT_ORD CAT_PT &out_vars_list.);
    by PAGE_ORD CAT_ORD TOGGLE PT_ORD;
  run;


  %skip_calc:;


  %if &data_rec_num. = 0 %then %do;
    data report_dsin;
      format PAGE_ORD CAT_ORD TOGGLE PT_ORD CAT_PT
&out_vars_list.;
      length TOGGLE CAT_PT &out_vars_list. $200 PAGE_ORD
CAT_ORD PT_ORD 8;
```

```
      if 0 then call missing(PAGE_ORD, CAT_ORD, TOGGLE,
PT_ORD, CAT_PT, %sysfunc(tranwrd(&out_vars_list., %str( ),
%str(, ))));
    run;
  %end;

  data &dsout. (keep = CAT_PT &out_vars_list.);
    set report_dsin;
    if &data_rec_num. ^= 0 then output;
  run;

  %if %upcase(&do_cleanup.) = TRUE %then %do;
    proc sql noprint;
      drop table adsl_data, input_data, trtgrp_names,
trtgrp_names_tr, trtgrp_vnames, sl_input_data, fmt_ds;
      %if &data_rec_num. ^= 0 %then %do;
        drop table subj_count, subj_count_c, subj_count_c_srt,
subj_count_c_tr, percents, percents_ntot, percents_ntot_srt,
cat_ord, cat_pt_ord, cat_ord_srt;
        %if %upcase(&do_calc_evt.) = TRUE %then %do;
          drop table evnt_count, evnt_count_c,
evnt_count_c_srt, evnt_count_c_tr;
        %end;
      %end;
    quit;
  %end;

  %if %upcase(&do_out_info.) = TRUE %then %do;
    %let length_max = 15;
```

```
    %do i = 1 %to &trt_cnt.;
      %if %eval(&length_max. < %length(%scan(&trt_names., &i.,
|))) %then %let length_max = %length(%scan(&trt_names., &i.,
|));
    %end;
    %let out_log_text = %sysfunc(putc(Treatment Group,
&length_max..)) | Subjects (N=) | Order | Sorted by;
    %put;
    %put NOTE- ------------------------------------------------
----;
    %put NOTE: &macro_name. Treatment Groups Info:;
    %put NOTE- Number of treatment groups: &trt_cnt.;
    %put NOTE- &out_log_text.;
    %do i = 1 %to &trt_cnt.;
      %let out_log_text = %sysfunc(putc(%scan(&trt_names.,
&i., |), &length_max..)) | %sysfunc(putc(%scan(&trt_totals.,
&i., |), 13.)) | %sysfunc(putc(%scan(&trt_ords., &i., |), 5.))
|;
      %if &out_sort_varname. = SUBJ_TRT%scan(&trt_ords., &i.,
|) %then %let out_log_text = &out_log_text. %str(+);
      %put NOTE- &out_log_text.;
    %end;
    %put NOTE- ------------------------------------------------
----;
    %put;
  %end;

  %if %upcase(&do_out_report.) = TRUE %then %do;
    ods listing close;
```

```
    ods &rtf_pdf. file = "&out_file.";
    ods escapechar = '^';

    %let out_trt_col_width =
%sysfunc(round(%sysevalf(%sysfunc(max(&decimals_num. + 8,
%length(Subjects)))*2.86/46 + %sysevalf(2*&hor_margin. +
0.02)), 0.01));
    proc report data = report_dsin split = '*' missing
headskip nowd style(report) = {width = 100%} style(column) =
{&out_style.} style(header) = {&out_style. just = center};
      column PAGE_ORD CAT_ORD TOGGLE PT_ORD CAT_PT
        %do i = 1 %to &trt_cnt. -
%eval(%upcase(&do_calc_total.) ^= TRUE);
          ("%scan(&trt_names., &i., |)^n(N=%scan(&trt_totals.,
&i., |))"
          %if %upcase(&do_calc_evt.) = TRUE %then
%scan(&out_vars_list., 2*&i. - 1, %str( ))
%scan(&out_vars_list., 2*&i., %str( )));
          %else %scan(&out_vars_list.,   &i.,     %str( )));
        %end;
      ;

      define PAGE_ORD /order order = data noprint;
      define CAT_ORD  /order order = data noprint;
      define TOGGLE   /group order = data noprint;
      define PT_ORD   /order order = data noprint;
      define CAT_PT   /group order = data "&cat_header_text."
style(column) = {width = 4.0in} style(header) = {just = left};
```

```
      %do i = 1 %to &trt_cnt. - %eval(%upcase(&do_calc_total.)
^= TRUE);
        %if %upcase(&do_calc_evt.) = TRUE %then %do;
          define %scan(&out_vars_list., 2*&i. - 1, %str( ))
/display order = data "Subjects" style(column) = {width =
&out_trt_col_width.in} style(header) = {just = left};
          define %scan(&out_vars_list., 2*&i.,     %str( ))
/display order = data "Events"   style(column) = {width =
&out_trt_col_width.in} style(header) = {just = left};
        %end;
        %else %do;
          define %scan(&out_vars_list.,   &i.,     %str( ))
/display order = data "Subjects" style(column) = {width =
&out_trt_col_width.in} style(header) = {just = left};
        %end;
      %end;

      break   after  PAGE_ORD /page;
      compute before TOGGLE   /style = {&out_style.};
        if TOGGLE = CAT_PT then TOGGLE = '';
        line TOGGLE $200.;
      endcomp;
      %if &data_rec_num. = 0 %then %do;
        compute before CAT_PT /style = {&out_style. just =
center};
          line "&nodata_string_text.";
        endcomp;
      %end;
    run;
```

```
    ods &rtf_pdf. close;
    ods listing;
  %end;
%mend SOC_PT;
```