Towards Agreed-Upon Dose Tables. A Method for Deriving Dosage Tiers from Collected Data

Introduction

The goal is to create meaningful dosage tiers for each substance and method of administration, providing users with guidance on expected effects at different dosage levels.

The levels are defined as:

- Threshold: Minimum dosage where effects start to be noticeable.
- Light: Mild dosage.
- Common: Dosage where typical effects are felt by most users.
- Strong: Potent dosage with stronger effects.
- Heavy: Intense dosage beyond which harm may outweigh benefits.

Each tier is defined by a dosage range, ensuring non-overlapping intervals for clarity and interpretability. Confidence intervals are calculated for percentile-based thresholds to account for variability in the data.

Data Loading and Preprocessing

Data Fields

Each dose record includes:

- Substance: Name of the substance.
- Method: Method of administration (e.g., oral, inhalation).
- Amount: Dosage amount.
- Units: Units of measurement (e.g., mg, µg, g, ml).

Data Filtering

- Valid Units: Only records with units in `["mg", "µg", "g", "ml"]` are considered.
- Amount Standardization: Dosage amounts are standardized if necessary (e.g., converting all units to milligrams), but in the provided code, amounts are directly used without unit conversion.

Outlier Detection

Outliers are detected using the Modified Z-Score method, which is robust against non-normal distributions.

Steps:

1. Exclude Non-Positive Values: Only positive dosage amounts are considered.
2. Compute Median ((\tilde{X})):

$$\tilde{X} = \text{median}(X)$$

3. Compute Median Absolute Deviation (MAD):

$$\text{MAD} = \text{median}(|X_i - \tilde{X}|)$$

4. Compute Modified Z-Score for Each Observation:

$$Z_i = 0.6745 \times \frac{X_i - \tilde{X}}{\text{MAD}}$$

5. Threshold: Observations with (|Z_i| > 3.5) are considered outliers.

Note: The constant (0.6745) rescales the MAD to be consistent with the standard deviation for a normal distribution.

Dosage Tier Calculation

Percentile-Based Thresholds

Dosage tiers are defined based on specific percentiles of the dosage amounts to ensure non-overlapping ranges.

Selected Percentiles:

- (P_5): 5th percentile ((0.05))
- (P_{25}): 25th percentile ((0.25))
- (P_{50}): 50th percentile ((0.50))
- (P_{75}): 75th percentile ((0.75))
- (P_{95}): 95th percentile ((0.95))

Tier Definitions:

1. Threshold: From minimum amount to (P_5)
2. Light: Greater than (P*5) up to (P*{25})
3. Common: Greater than (P*{25}) up to (P*{75})
4. Strong: Greater than (P*{75}) up to (P*{95})
5. Heavy: Greater than (P_{95}) up to maximum amount

```python
def compute_dose_tiers(group):
    amounts = group['amount_standard'].dropna().values
    if len(amounts) < 15:
        return pd.Series(dtype='float64')

    results = compute_dose_tiers_cy(amounts)

    min_amount = amounts.min()
    max_amount = amounts.max()

    labels = ['Threshold', 'Light', 'Common', 'Strong', 'Heavy']
    result = {}

    thresholds = [min_amount, results[0], results[3], results[6], results[9], max_amount]
    ci_lowers = [np.nan, results[1], results[4], results[7], results[10], np.nan]
    ci_uppers = [np.nan, results[2], results[5], results[8], results[11], np.nan]

    for i, label in enumerate(labels):
        result[f'{label} Lower'] = thresholds[i]
        result[f'{label} Upper'] = thresholds[i+1]
        result[f'{label} CI Lower'] = ci_lowers[i]
        result[f'{label} CI Upper'] = ci_uppers[i]

    reliability_score = calculate_reliability_score(group)
    result['Reliability Score (0 to 1)'] = reliability_score
    result['Unit'] = group['units'].iloc[0]
    return pd.Series(result)
```

Rationale for Percentile Choices

- Balanced Distribution: Using percentiles like (P*{25}) and (P*{75}) helps to capture the middle 50% of the data in the Common tier.
- Tail Coverage: (P*5) and (P*{95}) capture the lower and upper tails, ensuring extreme dosages are appropriately categorized.

Confidence Interval Calculation

To account for variability in the data and provide an estimate of the uncertainty associated with each percentile threshold, confidence intervals are calculated. The method used depends on the sample size:

- Sample Size (n < 75): Bootstrapping is used to estimate confidence intervals.
- Sample Size (n \geq 75): The Beta distribution method is used for analytical confidence intervals.

Bootstrapping Method (for N < 75)

Purpose

Bootstrapping is a non-parametric method that estimates the sampling distribution of a statistic by repeatedly resampling with replacement from the observed data.

Steps:

1. Resample: Generate multiple bootstrap samples by resampling the original data with replacement.
2. Compute Percentiles: For each bootstrap sample, compute the desired percentiles.
3. Construct Confidence Intervals: Determine the confidence intervals by taking percentiles (e.g., 2.5th and 97.5th) of the bootstrap distribution of the percentile estimates

Beta Distribution Method (for N >= 75)

Purpose

For larger sample sizes, the distribution of the sample percentiles can be approximated using the Beta distribution, allowing for analytical calculation of confidence intervals.

Steps:

1. Determine Parameters:
   - For a given percentile (p), the rank (k) is:

$$k = \lceil (n+1) \times p \rceil$$

2. Compute Confidence Bounds:
   - The lower and upper confidence bounds for the cumulative distribution function (CDF) are computed using the Beta distribution's inverse cumulative density function (CDF), denoted as (\text{Beta}^{-1}).
   - Lower bound ((\alpha) is the significance level, e.g., (\alpha = 0.05) for 95% confidence interval):

$$L = \text{Beta}^{-1}\left(\frac{\alpha}{2}, k, n-k+1\right)$$

   - Upper bound:

$$U = \text{Beta}^{-1}\left(1-\frac{\alpha}{2}, k, n-k+1\right)$$

3. Determine Indices:
   - Lower index:

$$i_L = \max\left(0, \lfloor L \times n \rfloor - 1\right)$$

   - Upper index:

$$i_U = \min\left(n-1, \lceil U \times n \rceil - 1\right)$$

4. Obtain Confidence Interval Values:

   - The confidence interval is given by:

$$\left[X_{(i_L)},\ X_{(i_U)}\right]$$

   Where (X_{(i)}) denotes the (i)-th ordered observation.

```
@cython.boundscheck(False)
@cython.wraparound(False)
def calculate_percentile_confidence_interval_cy(np.ndarray[DTYPE_t, ndim=1] data, DTYPE_t percentile, DTYPE_t alpha=0.05):
    """
    Calculates confidence intervals for a percentile using the beta distribution method.
    """
    cdef int n = data.shape[0]
    cdef int k
    cdef DTYPE_t lower_bound, upper_bound
    cdef int lower_index, upper_index

    if n == 0:
        return np.nan, np.nan

    data.sort()  # In-place sort

    k = int(np.ceil((n + 1) * percentile))
    from scipy.stats import beta
    lower_bound = beta.ppf(alpha / 2, k, n - k + 1)
    upper_bound = beta.ppf(1 - alpha / 2, k, n - k + 1)

    lower_index = max(0, int(np.floor(lower_bound * n)) - 1)
    upper_index = min(n - 1, int(np.ceil(upper_bound * n)) - 1)

    return data[lower_index], data[upper_index]
```

Combined Implementation in Cython

The `compute_dose_tiers_cy` function combines both methods, selecting the appropriate confidence interval calculation based on the sample size.

```
@cython.boundscheck(False)
@cython.wraparound(False)
def compute_dose_tiers_cy(np.ndarray[DTYPE_t, ndim=1] amounts):
    """
    Main function to compute dose tiers with adjusted percentiles for non-overlapping ranges.
    Uses bootstrapping for small samples and the Beta distribution method for larger samples.
    """
    cdef int n = amounts.shape[0]
    cdef np.ndarray[DTYPE_t, ndim=1] percentiles = np.array([0.05, 0.25, 0.50, 0.75, 0.95], dtype=np.float64)
    cdef int num_percentiles = percentiles.shape[0]  # num_percentiles = 5
    cdef np.ndarray[DTYPE_t, ndim=1] results = np.zeros(num_percentiles * 3, dtype=np.float64)  # Size 15
    cdef int i
    cdef np.ndarray[DTYPE_t, ndim=1] ci_lower = np.zeros(num_percentiles, dtype=np.float64)
    cdef np.ndarray[DTYPE_t, ndim=1] ci_upper = np.zeros(num_percentiles, dtype=np.float64)
    cdef np.ndarray[DTYPE_t, ndim=1] original_percentiles

    if n < 15:
        return np.array([np.nan] * (num_percentiles * 3))

    amounts.sort()  # Sort in-place

    if n < 75:
        # Use bootstrapping for confidence intervals
        original_percentiles, ci_lower, ci_upper = bootstrap_percentiles(amounts, percentiles)
    else:
        # Use Beta distribution method for confidence intervals
        original_percentiles = np.percentile(amounts, percentiles * 100)
        for i in range(num_percentiles):
            ci_lower[i], ci_upper[i] = calculate_percentile_confidence_interval_cy(amounts, percentiles[i])

    # Populate the results array with the percentile boundaries and confidence intervals
    for i in range(num_percentiles):
        results[i * 3] = original_percentiles[i]
        results[i * 3 + 1] = ci_lower[i]
        results[i * 3 + 2] = ci_upper[i]

    return results
```

Reliability Score Calculation

Purpose

To assess the reliability of the computed dosage tiers based on the quality and consistency of the data.

Components

1. Sample Size Score ((S_s)):

$$S_s = \min\left(1, \frac{n}{n_{\max}}\right)$$

Where:

- n = Sample size of the group.

- n_max = Maximum sample size considered optimal (set to 15).

2. Consistency Score ((S_c)):

$$\text{CV} = \frac{\sigma}{\mu}$$

$$S_c = \max(0, 1 - \text{CV})$$

Where:

- sigma = Standard deviation of dosage amounts.
- mu = Mean of dosage amounts.
- If mu = 0, CV = infinity.

3. Completeness Score ((S_p)):

$$S_p = \text{Proportion of non-missing dosage amounts}$$

Combined Reliability Score ((R))

$$R = 0.6 \times S_s + 0.3 \times S_c + 0.1 \times S_p$$

```python
def calculate_reliability_score(group):
    sample_size = len(group)
    max_sample_size = 15
    size_score = min(1, sample_size / max_sample_size)
    mean_amount = group["amount"].mean()
    std_amount = group["amount"].std()
    cv = std_amount / mean_amount if mean_amount != 0 else np.inf
    consistency_score = max(0, 1 - cv)
    completeness = group["amount"].notna().mean()
    reliability_score = (size_score * 0.6) + (consistency_score * 0.3) + (completeness * 0.1)
    return reliability_score
```

Rationale for Weights

- Sample Size ((60%)): Larger sample sizes provide more reliable estimates.
- Consistency ((30%)): Lower variability (consistency) in the data increases reliability.
- Completeness ((10%)): Data completeness ensures that missing values do not bias the results.

Additional Notes

Upper, Lower, CI Upper, CI Lower

The distinction between 'Lower/Upper' and 'CI Lower/CI Upper' in dosage tiers serves to differentiate between the estimated boundaries of each tier and the uncertainty that accompanies those estimates. The 'Lower' and 'Upper' limits delineate the specific dosage ranges for each tier, derived from calculated percentiles based on the sample data—these serve as the point estimates for dosage categorization. Conversely, the 'CI Lower' and 'CI Upper' limits denote the confidence intervals for these percentile estimates, reflecting the range within which the true population percentiles are expected to fall with a specified level of confidence (e.g., 95%). This differentiation informs us of both the precise thresholds for each tier and the reliability of these estimates.

Handling of `NaN` Values in Confidence Intervals

- Confidence intervals for the Threshold tier's lower bound and the Heavy tier's upper bound are `NaN` because they correspond to the minimum and maximum observed values, for which variability cannot be estimated using these methods.

Assumptions and Limitations

- Data Quality: The method assumes that the data is representative and that any measurement errors are random.
- Distribution Shape: Dosage amounts probably don't follow a normal distribution (bad to assume that in this context!); hence, non-parametric methods (percentiles, bootstrapping) and distribution-free methods (Beta distribution) are used.
- Unit Consistency: The method relies on the units being consistent within each group. Unit conversion is not explicitly handled in the code.

Possible Improvements

- Data Transformation: Applying a log transformation to dosage amounts could stabilize variance and make percentile estimation more robust in skewed distributions.

- **Unit Standardization**: Enforcing unit conversion to a standard unit (e.g., milligrams) would improve consistency across the dataset.
- **Dynamic Percentiles**: Adjusting percentile thresholds based on the distribution of each substance could provide more tailored dosage tiers.
