**Statistical Methods Document for Code Package**

This document describes the statistical methods and algorithms implemented in the code package. It focuses on three core techniques: **cosinor analysis**, **outlier detection**, and **zero amplitude testing**. These methods are commonly used in statistical analysis of periodic data, such as time series or biological rhythms.

**1. Cosinor Analysis**

**Overview**

Cosinor analysis is a statistical technique used to fit a cosine curve to periodic data, enabling the identification of rhythmic patterns. It is particularly useful in fields like chronobiology for analyzing circadian rhythms.

**Methodology**

The cosinor model assumes that the data can be represented by a cosine function of the form:

[ y(t) = M + A \cos(2\pi t / T + \phi) + e(t) ]

Where:

* ( y(t) ): Observed value at time ( t )
* ( M ): Mesor (midline estimating statistic of rhythm, or the mean level of the rhythm)
* ( A ): Amplitude (half the peak-to-trough difference of the rhythm)
* ( T ): Period (duration of one complete cycle, often fixed based on domain knowledge, e.g., 24 hours for circadian rhythms)
* ( \phi ): Acrophase (phase shift, indicating the time of the peak)
* ( e(t) ): Random error term

**Algorithm**

1. **Parameter Estimation**: The parameters ( M ), ( A ), and ( \phi ) are estimated using least squares regression. The cosine function is linearized by expressing it as: [ y(t) = M + \beta\_1 \cos(2\pi t / T) + \beta\_2 \sin(2\pi t / T) + e(t) ] where ( \beta\_1 = A \cos(\phi) ) and ( \beta\_2 = -A \sin(\phi) ).
2. **Fit the Model**: Solve the regression to obtain ( M ), ( \beta\_1 ), and ( \beta\_2 ).
3. **Derive Amplitude and Acrophase**: Compute ( A = \sqrt{\beta\_1^2 + \beta\_2^2} ) and ( \phi = \arctan(-\beta\_2 / \beta\_1) ), adjusting for quadrant.
4. **Statistical Testing**: Assess the significance of the rhythm by testing if ( A ) differs from zero (see Zero Amplitude Testing below).

**2. Outlier Detection**

**Overview**

Outlier detection identifies data points that deviate significantly from the expected pattern, which is critical for ensuring the reliability of statistical models like cosinor analysis.

**Methodology**

The code package employs a **z-score-based approach** to detect outliers in the residuals of the fitted model.

**Algorithm**

1. **Fit the Model**: Apply cosinor analysis (or another model) to the data and compute residuals ( e(t) = y(t) - \hat{y}(t) ), where ( \hat{y}(t) ) is the predicted value.
2. **Calculate Z-Scores**: For each residual, compute the z-score: [ z\_i = \frac{e\_i - \mu\_e}{\sigma\_e} ] where ( \mu\_e ) is the mean of the residuals (ideally 0 for a good fit), and ( \sigma\_e ) is the standard deviation.
3. **Thresholding**: Flag data points with ( |z\_i| > 3 ) (or a user-defined threshold) as outliers, indicating they are more than 3 standard deviations from the mean.
4. **Handling Outliers**: Options include removing outliers and refitting the model or reporting them for further investigation.

**3. Zero Amplitude Testing**

**Overview**

Zero amplitude testing determines whether a detected rhythm is statistically significant by testing the null hypothesis that the amplitude ( A = 0 ) (i.e., no rhythm exists).

**Methodology**

This test is based on the regression coefficients from the cosinor analysis and uses an **F-test** to assess the joint significance of the cosine and sine terms.

**Algorithm**

1. **Null Hypothesis**: ( H\_0: A = 0 ) (equivalent to ( \beta\_1 = 0 ) and ( \beta\_2 = 0 )).
2. **Fit Two Models**:
   * Reduced model: ( y(t) = M + e(t) ) (no rhythm).
   * Full model: ( y(t) = M + \beta\_1 \cos(2\pi t / T) + \beta\_2 \sin(2\pi t / T) + e(t) ).
3. **Compute F-Statistic**: [ F = \frac{(SSR\_{\text{reduced}} - SSR\_{\text{full}}) / 2}{SSR\_{\text{full}} / (n - 3)} ] where ( SSR ) is the sum of squared residuals, and ( n ) is the number of observations.
4. **Decision Rule**: Compare the F-statistic to a critical value from the F-distribution with 2 and ( n-3 ) degrees of freedom. If ( p < 0.05 ) (or a user-defined threshold), reject ( H\_0 ), indicating a significant rhythm.

**Conclusion**

The code package integrates these methods to analyze periodic data effectively:

* **Cosinor Analysis** fits rhythmic patterns and extracts key parameters.
* **Outlier Detection** ensures data quality by identifying anomalies.
* **Zero Amplitude Testing** validates the presence of significant rhythms.

These techniques are implemented with flexibility, allowing users to adjust parameters such as period length or outlier thresholds based on their specific needs.

### Circadian Rhythmicity Analysis R Package: Code Flow Description

This R code package is designed to analyze circadian rhythms in neurocompound concentration data over Zeitgeber Time (ZT), providing a streamlined pipeline from raw data input to final output. The package consists of a series of modular scripts that handle data preprocessing, modeling, analysis, visualization, and result export. This writeup describes how the code progresses as a cohesive system, detailing the sequence of operations and the interactions between its components.

The workflow begins with raw data in a data frame format, typically containing two key columns: ZT (Zeitgeber Time, ranging from 0 to 23) and Concentration(mg) (neurocompound concentration measurements). The code then processes this data through a series of steps, each implemented in a dedicated script, culminating in graphical and tabular outputs that summarize the circadian characteristics of the data.

#### 1. Data Preprocessing: Outlier Detection and Cleaning

The analysis starts with data cleaning, handled by two scripts: outlier\_detection.R and filter\_outliers.R. The outlier\_detection.R script scans the numeric columns of the input data frame—primarily focusing on Concentration(mg)—and applies the Interquartile Range (IQR) method to identify outliers. It calculates the first (Q1) and third (Q3) quartiles, defines the IQR as Q3 - Q1, and flags values outside the range [Q1 - 1.5IQR, Q3 + 1.5IQR] as outliers. This script outputs a logical structure indicating which values are outliers.

Next, filter\_outliers.R takes the original data frame and the outlier detection output, replacing flagged outlier values with NA. This step ensures that subsequent modeling steps operate on a cleaned dataset, free from extreme values that could skew the results. The cleaned data frame retains its original structure, with NA values marking the excluded points, allowing seamless integration into the next phase.

#### 2. Cosinor Model Fitting

With the data cleaned, the package fits a cosinor model to capture the sinusoidal pattern of circadian rhythms. This step relies on an external library, cosinor2, which must be loaded into the R environment. The core operation occurs within a single function call, typically structured as cosinor.lm() in the workflow. This function takes the cleaned data frame, specifies the dependent variable (Concentration(mg)), and maps it against the independent time variable (ZT) with a fixed 24-hour period.

The cosinor model assumes a sinusoidal form, ( y = M + A \cdot \cos(2\pi t / 24 + \phi) ), where ( M ) is the mesor (mean level), ( A ) is the amplitude, and ( \phi ) is the acrophase (phase shift). The output is a fitted model object containing coefficients and statistical metrics, which serves as the foundation for all subsequent analyses. This step bridges the raw data to the analytical outputs by encoding the rhythmic behavior mathematically.

#### 3. Parameter Extraction

The fitted model is then processed by extract\_cosinor.R, which extracts key circadian parameters from the cosinor fit. This script parses the model object to compute the mesor, amplitude, acrophase (in both radians and hours), and bathyphase (trough time, derived as acrophase + 12 hours). It also calculates standard deviations and errors for amplitude and acrophase, providing measures of uncertainty.

The script organizes these results into two formats: a data frame (results\_table) for raw data storage and a graphical table (results\_grob) using the grid package for visual presentation. These outputs are bundled into a list, making them accessible for both further processing and direct export. This step transforms the abstract model into concrete, interpretable metrics that describe the rhythm’s properties.

#### 4. Zero Amplitude Testing

To assess the statistical significance of the detected rhythm, zero\_amp.R performs a zero amplitude test on the fitted model. This script conducts an F-test to determine whether the amplitude is significantly different from zero, indicating a true circadian pattern rather than random fluctuation. It extracts degrees of freedom, the F-value, and the p-value from the model, with a p-value threshold (typically < 0.05) signaling significance.

Similar to extract\_cosinor.R, it produces a list containing a data frame and a graphical table, ensuring consistency in output format across the package. This step integrates with the parameter extraction by adding a layer of statistical validation, linking the model’s descriptive outputs to a hypothesis-testing framework.

#### 5. Visualization

The plot\_cosinor.R script visualizes the analysis by plotting the original data points (from the cleaned data frame) alongside the fitted cosinor curve. It generates a scatter plot with the Concentration(mg) values against ZT, overlaying a smooth sinusoidal line derived from the model’s predictions. The script allows customization of the plot’s title and styling (e.g., point colors and line types) based on a data\_type parameter, which distinguishes between conditions like "LD" (light-dark) or "rLEN" (other experimental setups).

This visualization step ties together the preprocessing, modeling, and analysis phases by providing a graphical representation of how well the model captures the data’s rhythmic behavior. The plot object is returned as a ggplot2 object, ready for display or export.

#### 6. Result Export

Finally, the package consolidates its outputs through two export scripts: export\_excel.R and create\_combined\_pdf.R. The export\_excel.R script scans the R environment for graphical table objects (grobs) produced by extract\_cosinor.R and zero\_amp.R, matching them by a user-specified prefix (e.g., "experiment1"). It then writes each grob to a separate sheet in an Excel file using an Excel-writing library like openxlsx, preserving the formatted tables for easy sharing.

The create\_combined\_pdf.R script takes a more structured approach, accepting a list of up to six grob objects along with headers and a title. It arranges these tables in a grid layout on a single PDF page, using the grid package to manage the layout, and saves the result to a specified file path. This script integrates the tabular outputs from the parameter extraction and testing steps into a polished, report-ready format.

#### Overall Code Flow

The code progresses linearly through these steps, with each script building on the output of the previous one:

* **Input Data** → outlier\_detection.R → filter\_outliers.R → Cleaned Data
* **Cleaned Data** → cosinor.lm() → Fitted Model
* **Fitted Model** → extract\_cosinor.R → Parameters (Tables/Grobs)
* **Fitted Model** → zero\_amp.R → Test Results (Tables/Grobs)
* **Cleaned Data + Fitted Model** → plot\_cosinor.R → Plot
* **Tables/Grobs** → export\_excel.R or create\_combined\_pdf.R → Exported Files

The modular design ensures that intermediate outputs (e.g., cleaned data, fitted model, parameter tables) can be inspected or reused, while the consistent list-based output structure from analysis scripts facilitates integration into the export phase. The package assumes the presence of external dependencies (cosinor2, ggplot2, grid, etc.), which must be installed prior to execution.

This flow encapsulates the full progression from raw neurocompound data to a comprehensive circadian rhythm analysis, delivering both quantitative insights and visual summaries in a systematic, reproducible manner.