# **DIanalyst: A Shiny App for End-to-End Metabolomics Data Preprocessing Focused on Direct-Injection Analysis**

John Lennon Calorio1, Gerard Bryan Gonzales2

1Center for Research and Development, Davao Medical School Foundation Inc., Davao City, Philippines

2Department of Public Health and Primary Care, Faculty of Medicine and Health Sciences, Ghent University, Ghent, Belgium

# **Abstract**

**Motivation:** Metabolomics data preprocessing involves multiple sequential steps including outlier removal, missing value imputation, signal drift and batch correction, data normalization, mathematical transformation, data scaling, and feature filtering. Current workflows often require manual coordination of multiple packages and careful parameter tuning, creating barriers for researchers and potential inconsistencies in data processing. A unified, automated preprocessing pipeline with built-in quality control would significantly improve reproducibility and accessibility in metabolomics research.

**Results:** We present DIanalyst, a Shiny-based R application that integrates the complete direct-injection metabolomics preprocessing workflow into a single, automated pipeline. The function processes data from the quality check stage through to analysis-ready matrices, handling outlier removal, missing value imputation using minimum fraction replacement, Quality Control-Robust Spline Correction (QC-RSC) signal drift and batch correction with uncorrected feature detection and removal option, multiple normalization methods (sum, median, PQN, quantile), various transformation approaches including VSN, Pareto and unit variance scaling, and RSD-based and variance-based feature filtering. The app provides extensive parameter customization while maintaining sensible defaults based on current best practices, and returns comprehensive documentation of all processing steps with dimensional tracking and quality metrics.

**Availability:** The DIanalyst app is freely available at \_\_\_\_\_\_\_\_\_\_\_\_.

**Keywords:** metabolomics, data preprocessing, QC-RSC, normalization, signal drift and batch correction

# **1. Introduction**

Metabolomics data preprocessing is a critical step that significantly impacts downstream statistical analyses and biological interpretation. The typical workflow involves multiple sequential operations: outlier removal, missing value handling, signal drift and batch effect correction, data normalization, mathematical transformation, data scaling, and feature filtering based on analytical quality metrics. Each step requires careful consideration of method selection and parameter optimization, often necessitating expertise across multiple R packages and statistical approaches.

Current solutions typically address individual preprocessing steps in isolation, requiring researchers to manually coordinate multiple tools and maintain consistency across the workflow. Popular platforms like MetaboAnalyst provide web-based preprocessing but lack the flexibility needed for complex experimental designs, while individual R packages like *pmp* focus on specific aspects of the preprocessing pipeline. This fragmented approach can lead to inconsistencies, suboptimal parameter selection, and reduced reproducibility.

The DIanalyst web-app addresses these limitations by providing a comprehensive, automated preprocessing pipeline that integrates established methods from multiple packages into a single, cohesive workflow while maintaining full parameter control and extensive documentation of processing decisions.

# **2. Implementation**

In this section, we illustrate how DIanalyst web-app works and how to use it. The documentation and the step-by-step tutorial are available at \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

## **2.1 Function Architecture and Data Flow**

The DIanalyst web-app is designed as a sequential pipeline that processes data from the output of a quality check function through to analysis-ready matrices.

The pipeline consists of eight major processing stages executed in a fixed sequence to ensure optimal data quality:

1. Input Validation and Data Preparation
2. Outlier Removal and Metadata Extraction
3. Missing Value Filtering and Imputation
4. Signal Drift and Batch Effect Correction
5. Data Normalization
6. Mathematical Transformation
7. Data Scaling
8. Quality-Based Feature Filtering

## **2.2 Core Parameters and Functionality**

### **2.2.1 Input Data and Outlier Management**

**raw\_data**: The primary input parameter accepting a list object from the *perform\_DataQualityCheck* function. The function performs extensive validation to ensure data structure integrity, including verification of the function origin, presence of required metadata fields, and dimensional consistency.

**outliers**: A vector specifying biological and/or QC samples to be removed before processing. The function provides comprehensive reporting of successfully removed outliers and samples not found in the dataset, ensuring transparency in outlier handling decisions.

### **2.2.2 Missing Value Handling Parameters**

The function implements a two-stage approach to missing value management: filtering of features with excessive missingness followed by imputation of remaining missing values.

**filterMissing**: Defines the minimum percentage threshold (1-100%) for missing values required to remove a feature from the dataset. Features exceeding this threshold across the specified sample groups are excluded from downstream processing.

**filterMissing\_by\_group**: Boolean parameter controlling whether missing value assessment considers group-specific patterns. When TRUE, features are removed only if they exceed the threshold in ALL sample groups, preserving features that may be informative for specific experimental conditions.

**filterMissing\_includeQC**: Boolean parameter determining whether QC samples are included in missing value threshold calculations. Setting to FALSE (default) focuses filtering on biological samples, preventing removal of features that may be absent in QC but present in experimental samples.

**denMissing**: Denominator value used in the fraction 1/denMissing for replacing remaining missing values with a fraction of the minimum detected value per feature. This approach preserves the relative relationships between features while avoiding arbitrary value assignment.

### **2.2.3 Drift and Batch Correction Parameters**

The function implements the QC-RSC (Quality Control-Robust Spline Correction) algorithm for systematic error correction, with advanced detection of features that could not be reliably corrected.

**driftBatchCorrection**: Boolean parameter controlling application of the QC-RSC algorithm. When TRUE, the function applies spline-based correction using QC sample trajectories to model and correct systematic variations.

**spline\_smooth\_param**: Numeric parameter (0-1) controlling the smoothing parameter for spline fitting. Values closer to 0 produce more flexible fits, while values closer to 1 produce smoother corrections.

**spline\_smooth\_param\_limit**: Vector of format c(min, max) defining the allowable range for automatic spline parameter optimization during correction.

**log\_scale**: Boolean parameter determining whether signal correction fitting is performed on log-scaled data, which is typically recommended for metabolomics data due to its multiplicative error structure.

**min\_QC**: Minimum number of QC samples required per batch for reliable correction. Features in batches with insufficient QC samples are flagged as uncorrectable.

**removeUncorrectedFeatures**: Boolean parameter controlling whether features that could not be corrected due to insufficient QC samples are automatically removed from the dataset. The function provides comprehensive reporting of uncorrected features regardless of this setting.

### **2.2.4 Normalization Parameters**

The function provides multiple normalization approaches to account for systematic differences in overall metabolite abundance between samples.

**dataNormalize**: String parameter specifying the normalization method with eight options:

* "none": No normalization applied
* "Normalization": Uses values from the "Normalization" metadata field when available
* "sum": Normalizes by total sample sum (suitable for compositional data)
* "median": Normalizes by sample median (robust to outlying features)
* "PQN1": Probabilistic Quotient Normalization using median reference spectrum
* "PQN2": PQN using a specific reference sample
* "groupPQN": Group-specific PQN using pooled QC samples as reference
* "quantile": Quantile normalization for complex batch designs

**refSample**: String parameter specifying the reference sample identifier for dataNormalize = "PQN2".

**groupSample**: String parameter used for dataNormalize = "groupPQN", specifying which QC sample types to use as reference (options: "SQC", "EQC", or "both").

**reference\_method**: String parameter controlling reference calculation method for quantile normalization ("mean" or "median").

### **2.2.5 Transformation Parameters**

**dataTransform**: String parameter specifying mathematical transformation applied after normalization:

* "none": No transformation
* "log2": Base-2 logarithm transformation
* "log10": Base-10 logarithm transformation
* "sqrt": Square root transformation
* "cbrt": Cube root transformation
* "vsn": Variance Stabilizing Normalization using the vsn package

The function automatically handles negative values and zeros through appropriate offsetting before log-based transformations.

### **2.2.6 Data Scaling Parameters**

The function provides separate scaling options optimized for different downstream analyses:

**dataScalePCA**: String parameter for PCA-optimized scaling:

* "none": No scaling applied
* "mean": Mean centering only
* "meanSD": Unit variance scaling (z-score standardization)
* "mean2SD": Pareto scaling (mean-centered, divided by square root of standard deviation)

**dataScalePLS**: String parameter for PLS-optimized scaling with the same options as dataScalePCA. Pareto scaling ("mean2SD") is typically recommended for PLS-based analyses.

### **2.2.7 Quality-Based Filtering Parameters**

**filterMaxRSD**: Numeric parameter defining the maximum allowed relative standard deviation (RSD) percentage in QC samples. Features exceeding this threshold are removed as they represent poor analytical precision.

**filterMaxRSD\_by**: String parameter specifying which QC samples to use for RSD calculation:

* "SQC": Sample QC (pooled biological samples)
* "EQC": Extract QC (technical replicates)
* "both": Combined SQC and EQC samples

**filterMaxVarSD**: Numeric parameter specifying the percentile threshold for removing features with lowest biological variability, helping to focus analyses on informative metabolites.

## **2.3 Algorithmic Workflow and Processing Pipeline**

The function implements a rigorous sequential processing workflow with comprehensive error handling and dimensional tracking:

**Stage 1: Data Validation and Preparation**

* Validates input data structure and origin
* Performs parameter validation with detailed error reporting
* Transposes data matrix and sorts by injection sequence
* Extracts and validates required metadata fields

**Stage 2: Outlier Management and Metadata Processing**

* Removes specified outlier samples with detailed reporting
* Creates comprehensive metadata structure with standardized group labels
* Establishes QC and non-QC sample indices for downstream processing

**Stage 3: Missing Value Processing**

* Converts zeros to NA values for proper missing value handling
* Applies group-aware missing value filtering with optional QC inclusion
* Implements minimum fraction imputation strategy preserving feature relationships
* Removes features with all missing values

**Stage 4: Drift and Batch Correction**

* Applies QC-RSC algorithm using the pmp package with error handling
* Implements advanced detection of uncorrected features through before/after comparison
* Provides detailed reporting of correction success and failure reasons
* Optional removal of uncorrected features with comprehensive documentation

**Stage 5: Data Normalization**

* Implements multiple normalization approaches with automatic fallback mechanisms
* Handles method-specific requirements (reference samples, QC groups)
* Provides detailed logging of applied normalization strategies

**Stage 6: Mathematical Transformation**

* Applies selected transformation with automatic handling of edge cases
* Implements VSN with fallback to simpler methods if package unavailable
* Ensures data compatibility with downstream analyses

**Stage 7: Data Scaling for Multiple Analyses**

* Creates separate scaled datasets optimized for PCA and PLS analyses
* Implements Pareto, unit variance, and mean centering approaches
* Maintains data integrity across different scaling strategies

**Stage 8: Quality-Based Feature Filtering**

* Applies RSD-based filtering using pmp package with error handling
* Implements variance-based filtering for biological relevance
* Creates final analysis-ready datasets with comprehensive dimensional tracking

## **2.4 Output Structure and Documentation**

The function returns a comprehensive list object containing:

* **Multiple processed data matrices** at each pipeline stage for transparency and flexibility
* **Complete parameter documentation** enabling full reproducibility
* **Dimensional tracking** showing sample and feature counts at each processing step
* **Quality metrics** including data completeness and feature retention rates
* **Processing summary** with detailed information about applied methods
* **Visualization plots** for drift/batch correction assessment
* **Uncorrected feature documentation** with detailed reasoning for correction failures
* **Warning and error documentation** for troubleshooting and method validation

# **3. Use Case and Implementation Example**

## **3.1 Standard Implementation**

library(MetaboStatR)

# Assume raw\_data comes from perform\_DataQualityCheck()

result <- perform\_PreprocessingPeakData(

raw\_data = quality\_checked\_data,

outliers = c("Sample\_001", "QC\_05"),

filterMissing = 20,

filterMissing\_by\_group = TRUE,

driftBatchCorrection = TRUE,

min\_QC = 5,

removeUncorrectedFeatures = TRUE,

dataNormalize = "sum",

dataTransform = "vsn",

dataScalePCA = "meanSD",

dataScalePLS = "mean2SD",

filterMaxRSD = 30,

filterMaxRSD\_by = "EQC",

filterMaxVarSD = 10

)

# Access different processed datasets

pca\_ready\_data <- result$data\_scaledPCA\_rsdFiltered\_varFiltered

pls\_ready\_data <- result$data\_scaledPLS\_rsdFiltered\_varFiltered

metadata <- result$Metadata

# Review processing summary

print(result$ProcessingSummary)

print(result$Dimensions)

## **3.2 Method Selection Guidelines**

Based on current metabolomics best practices, we recommend:

**Missing Value Handling:**

* filterMissing = 20-30% for discovery studies, 10-15% for targeted analyses
* filterMissing\_by\_group = TRUE to preserve condition-specific metabolites
* denMissing = 5 providing conservative imputation values

**Drift/Batch Correction:**

* driftBatchCorrection = TRUE for LC-MS data with >50 samples
* min\_QC = 5 ensuring reliable correction fitting
* removeUncorrectedFeatures = TRUE for stringent quality control

**Normalization:**

* "sum" normalization for general LC-MS metabolomics
* "median" normalization for robust handling of outlying features
* "PQN1" or "quantile" for complex experimental designs

**Transformation:**

* "vsn" for optimal variance stabilization in large datasets
* "log2" for interpretable fold-change analyses

**Scaling:**

* "mean2SD" (Pareto) for PLS analyses balancing high/low abundance metabolites
* "meanSD" (unit variance) for PCA when all features should contribute equally

**Filtering:**

* filterMaxRSD = 20-30% based on analytical platform precision
* filterMaxVarSD = 10% removing least variable features for focused analysis

# **4. Discussion**

## **4.1 Integration and Automation Benefits**

The *perform\_PreprocessingPeakData* function addresses critical gaps in metabolomics data processing by providing a unified, automated pipeline that maintains the flexibility required for diverse experimental designs. The integrated approach eliminates potential inconsistencies arising from manual coordination of multiple packages while preserving full parameter control for method optimization.

The function's comprehensive documentation and dimensional tracking enable full reproducibility and support transparent reporting of preprocessing decisions. The automatic detection and reporting of uncorrected features in drift/batch correction represents a significant advancement in quality control, as traditional implementations often fail to identify features that could not be reliably corrected.

## **4.2 Quality Control and Validation Features**

The function incorporates several advanced quality control features not commonly available in existing tools. The sophisticated missing value filtering considers group-specific patterns, preventing removal of metabolites that may be condition-specific. The drift correction module provides detailed assessment of correction success, including automatic detection of features with insufficient QC coverage for reliable correction.

The dual scaling approach generates separate datasets optimized for different downstream analyses (PCA vs. PLS), recognizing that optimal preprocessing parameters may differ based on the intended statistical approach. This eliminates the need for researchers to manually create multiple processed datasets for comprehensive analyses.

## **4.3 Computational Efficiency and Scalability**

The function is optimized for typical metabolomics datasets ranging from dozens to hundreds of samples with hundreds to thousands of features. Processing typically completes within minutes for most datasets, with automatic memory management preventing resource exhaustion in large-scale analyses.

Error handling and parameter validation prevent common user errors while providing informative feedback for troubleshooting. The modular internal architecture allows for easy maintenance and extension of individual processing steps.

## **4.4 Comparison to Existing Tools**

Unlike web-based platforms that limit parameter control and dataset size, this function provides complete customization while maintaining accessibility through sensible defaults. Compared to package-specific solutions that address individual preprocessing steps, the integrated approach ensures consistency and compatibility across all processing stages.

The comprehensive output documentation exceeds most existing tools by providing detailed tracking of processing decisions, quality metrics, and potential issues, supporting both novice users and experienced researchers requiring detailed method validation.

# **5. Conclusion**

The *perform\_PreprocessingPeakData* function provides a robust, comprehensive solution for metabolomics data preprocessing that addresses current limitations in available tools. By integrating multiple processing steps into a single, well-documented pipeline with extensive quality control features, it supports both standardization of preprocessing workflows and customization for specific experimental requirements.

The function's advanced features, including sophisticated missing value handling, automatic detection of uncorrected features, and dual-analysis optimization, represent significant improvements over existing approaches. The comprehensive documentation and quality tracking support reproducible research and transparent method reporting.

This implementation facilitates adoption of best-practice preprocessing approaches while reducing the technical barriers that often limit the accessibility of advanced metabolomics data analysis.

# **Availability and Implementation**

The *perform\_PreprocessingPeakData* function is implemented in R and freely available as part of the MetaboStatR package at <https://github.com/jllcalorio/MetaboStatR>. The function requires the pmp, vsn, ggplot2, tidyr, and gridExtra packages for full functionality, with automatic fallback mechanisms when optional packages are unavailable.

# **Acknowledgments**

# **References**

[References would include citations for QC-RSC algorithm (pmp package), VSN transformation, PQN normalization methods, and relevant metabolomics preprocessing best practices]