The Hitchhiker's Guide to Statistical Analysis of Feature-based Molecular Networks from Non-Targeted Metabolomics Data

Abzer K. Pakkir Shah[1,2], Axel Walter[1,2,3], Filip Ottosson[4], Francesco Russo[4], Marcelo Navarro-Diaz[2], Judith Boldt[1,5], Jarmo-Charles J. Kalinski[1,6], Eftychia E. Kontou[1,7], James Elofson[8], Alexandros Polyzois[1,9], Carolina González-Marín[1,10], Shane Farrell[11,12], Marie R. Aggerbeck[1,13], Thapanee Pruksatrakul[1,14], Nathan Chan[15], Yunshu Wang[15], Magdalena Pöchhacker[1,16], Corinna Brungs[17], Beatriz Cámara[18], Andrés Mauricio Caraballo-Rodríguez[19], Andres Cumsille[18], Fernanda de Oliveira[20,21], Kai Dührkopp[22], Yasin El Abiead[23], Christian Geibel[2], Lana G. Graves[24,25], Martin Hansen[13], Steffen Heuckeroth[26], Simon Knoblauch[2], Anastasiia Kostenko[8], Mirte C. M. Kuijpers[27], Kevin Mildau[1,28,29], Stilianos Papadopoulos Lambidis[2], Paulo Wender Portal Gomes[23], Tilman Schramm[2,30], Karoline Steuer-Lodd[2,30], Paolo Stincone[2], Sibgha Tayyab[2], Giovanni Andrea Vitale[2], Berenike C. Wagner[2], Shipei Xing[23], Marquis T. Yazzie[8], Simone Zuffa[23,31], Martinus de Kruijff[32], Christine Beemelmanns[32,33], Hannes P. Link[2], Christoph Mayer[2], Justin J.J. van der Hooft[1,29,34], Tito Damiani[17], Tomáš Pluskal[17], Pieter C. Dorrestein[23,31], Jan Stanstrup[35], Robin Schmid[1,17], Mingxun Wang[1,15], Allegra T. Aron[1,8], Madeleine Ernst[4,#], Daniel Petras[1,2,30,#]

1. Virtual Multi-Omics Laboratory, The Internet
2. University of Tuebingen, Interfaculty Institute of Microbiology and Infection Medicine, Tübingen 72076, Germany
3. Applied Bioinformatics, Department of Computer Science, University of Tübingen
4. Section for Clinical Mass Spectrometry, Danish Center for Neonatal Screening, Department of Congenital Disorders, Statens Serum Institut, Artillerivej 5, DK-2300 Copenhagen S, Denmark
5. Leibniz-Institute DSMZ - German Collection of Microorganisms and Cell Cultures, Braunschweig, Germany
6. Department of Biochemistry and Microbiology, Rhodes University, 6140, Makhanda, South Africa
7. The Novo Nordisk Foundation for Biosustainability, Technical University of Denmark, Kemitorvet 220, 2800 Kongens Lyngby, Denmark
8. Department of Chemistry and Biochemistry, University of Denver, USA
9. Boyce Thompson Institute and Department of Chemistry and Chemical Biology, Cornell University, Ithaca, New York 14853, USA
10. Universidad EAFIT, Carrera 49, Cl. 7 Sur #50, Medellín, Antioquia, Colombia
11. Bigelow Laboratory for Ocean Sciences, East Boothbay, ME 04544, USA
12. School of Marine Sciences, Darling Marine Center, University of Maine, Walpole, ME 04573, USA
13. Department of Environmental Science, Aarhus University, Frederiksborgvej 399, 4000 Roskilde, Denmark
14. National Center for Genetic Engineering and Biotechnology (BIOTEC), National Science and Technology Development Agency (NSTDA), Thailand Science Park, Pathum Thani, 12120, Thailand

15. University of California Riverside, Department of Computer Science, 900 University Ave, Riverside CA, USA
16. Department of Food Chemistry and Toxicology, University of Vienna, Vienna, Austria
17. Institute of Organic Chemistry and Biochemistry of the Czech Academy of Sciences, Prague, Czech Republic
18. Laboratorio de Microbiología Molecular y Biotecnología Ambiental, Centro de Biotecnología DAL, Universidad Técnica Federico Santa María, Valparaíso, Chile
19. Skaggs School of Pharmacy and Pharmaceutical Sciences, University of California, San Diego, 9500 Gilman Dr., San Diego, CA, 92093, USA
20. Department of Biotechnology, Engineering School of Lorena, University of São Paulo, Lorena, São Paulo 12602-810, Brazil
21. Skaggs School of Pharmacy and Pharmaceutical Sciences, University of California, San Diego, La Jolla, California 92093, USA
22. University of Jena, Department of Bioinformatics, Jena, Germany
23. Skaggs School of Pharmacy and Pharmaceutical Sciences, University of California San Diego, 9500 Gilman Dr., San Diego, CA, 92093, USA
24. University of Tuebingen, Department of Environmental Systems Analysis, Tübingen 72076, Germany
25. Leibniz Institute of Freshwater Ecology and Inland Fisheries (IGB), Berlin, Germany
26. Institute of Inorganic and Analytical Chemistry, University of Münster, Münster, Germany
27. Department of Ecology, Behaviour and Evoloution, University of California San Diego, 9500 Gilman Dr., San Diego, CA, 92093, USA
28. Department of Analytical Chemistry, University of Vienna, Austria
29. Bioinformatics Group, Wageningen University & Research, 6708 PB Wageningen, the Netherlands
30. University of California Riverside, Department of Biochemistry, 900 University Ave, Riverside, CA, USA
31. Collaborative Mass Spectrometry Innovation Center, Skaggs School of Pharmacy and Pharmaceutical Sciences, University of California San Diego, 9500 Gilman Dr., San Diego, CA, 92093, USA
32. Helmholtz Institute for Pharmaceutical Research Saarland (HIPS), Helmholtz Centre for Infection Research (HZI), Campus E8, 66123 Saarbrücken, Germany
33. Saarland University, Saarbrücken 66123, Germany
34. Department of Biochemistry, University of Johannesburg, Johannesburg 2006, South Africa
35. Department of Nutrition, Exercise and Sports, University of Copenhagen, 1958 Frederiksberg C, Denmark

Correspondence should be addressed to Madeleine Ernst (MAET@ssi.dk) or Daniel Petras (daniel.petras@uni-tuebingen.de)

Feature-Based Molecular Networking (FBMN) is a popular analysis approach for LC-MS/MS-based non-targeted metabolomics data. While processing LC-MS/MS data through FBMN is fairly streamlined, downstream data handling and statistical interrogation is often a key bottleneck. Especially, users new to statistical analysis struggle to effectively handle and analyze complex data matrices. In this protocol, we provide a comprehensive guide for the statistical analysis of FBMN results. We explain the data structure and principles of data clean-up and normalization, as well as uni- and multivariate statistical analysis of FBMN results. We provide explanations and code in two scripting languages (R and Python) as well as the QIIME2 framework for all protocol steps, from data clean-up to statistical analysis. Additionally, the protocol is accompanied by a web application with a graphical user interface (https://fbmn-statsguide.gnps2.org/), to lower the barrier of entry for new users. Together, the protocol, code, and web app provide a complete guide and toolbox for FBMN data integration, clean-up, and advanced statistical analysis, enabling new users to uncover molecular insights from their non-targeted metabolomics data. Our protocol is tailored for the seamless analysis of FBMN results from Global Natural Products Social Molecular Networking (GNPS and GNPS2) and can be adapted to other MS feature detection, annotation, and networking tools.

1. Introduction

Metabolomics aims to characterize and quantify the detectable spectrum of small molecules in order to catalog and understand the metabolic dynamics within biological systems[1,2]. Phenotypes or environmental factors that distinguish samples within a given set are often reflected in the chemical profiles of such small molecules across samples. Therefore, the characterization of chemical distinctions and gradients between samples provides crucial information for describing and understanding molecular mechanisms[3,4]. Metabolomics studies usually employ targeted or non-targeted approaches[2]. Targeted metabolomics is typically hypothesis-driven and aims to quantify known metabolites, often using internal standards and experimental methodology optimized for the study. Non-targeted metabolomics, on the other hand, aims to detect a maximum number of metabolites in order to comprehensively characterize the chemical profiles within a given sample set.

To uncover molecular insights from non-targeted liquid chromatography tandem mass spectrometry (LC-MS/MS) data, several software tools are available that assist with mining and annotating metabolites, including feature detection and annotation tools[5]. Feature-Based Molecular Networking (FBMN) is a popular analysis platform that integrates various feature-detection tools with molecular networking for metabolite annotation and annotation propagation[6] in the GNPS cloud ecosystem[7]. FBMN is routinely applied many biological disciplines, including clinical studies[8,9], plant[10–12] and environmental science[13–16], as well as microbiome[17–19] and the functional analysis of natural products[20–22]. While platforms such as GNPS have improved the way that we identify and characterize metabolites, the subsequent step — statistical analysis — remains a challenge for many researchers. While resources like MetaboAnalyst[23,24] provide powerful solutions for the statistical analysis of metabolomics data, the complex multi-layer information from FBMN results and other downstream annotation tools (e.g., SIRIUS) require typically multiple matrix operations, data clean-up, normalization, before uni- and multivariate statistical analyses. Most tools and analysis approaches that can be used to archive this are typically custom scripts or different software tools that are scattered across different platforms. This makes it especially difficult for users new to the field to effectively manage and analyze their data. Moreover, while there are several tools available for individual clean-up and analysis steps (see alternative approaches section), there is a lack of a comprehensive, user-friendly guidance that covers the entire spectrum of data preparation and statistical analysis of FBMN results.

In this protocol, we provide a detailed guide that starts with FBMN results, offering an end-to-end pipeline from feature detection, spectrum annotation, subsequent data clean-up and statistical analysis steps. This step-by-step guide is complemented with ready-made code for the popular statistical scripting and platforms R and Python, the QIIME2 toolkit (https://github.com/Functional-Metabolomics-Lab/FBMN-STATS), as well as a web application designed to simplify the process (https://fbmn-statsguide.gnps2.org/). The protocol provides thereby a seamless analysis guide for FBMN results from the GNPS (https://gnps.ucsd.edu), and GNPS2 (https://gnps2.org) web platforms, which can also be adapted to other MS feature detection and annotation tools.

Feature-based Molecular Networking from LC-MS/MS Data

Liquid chromatography-mass spectrometry (LC-MS) is one of the most prominent metabolomics techniques, with applications in numerous research fields[25–28]. Specifically, LC coupled with tandem mass spectrometry (LC-MS/MS) has been widely used because it provides a broad coverage of chemical space allowing for the simultaneous semi-quantitative detection and qualitative annotation of many metabolites over a wide dynamic range[14,29–32]. In addition to providing the molecular mass, retention time, and isotopic pattern of a metabolite, MS/MS provides structural information about the detected species. This is achieved through the fragmentation of precursor ions into product ions and the measurement of their mass-to-charge ratios (*m/z*) and abundances. This is usually done through Data-Dependent Acquisition (DDA), where ions that are observed in MS1 survey scans are iteratively selected for further fragmentation in subsequent MS/MS scans (See **Figure 1.1**). DDA operates by selecting the “top N” peaks in each duty cycle, where “N” is a user-defined number. These peaks are chosen based on their intensity and other user defined criteria through an automated process[15]. The resulting MS/MS spectra of product ions can be used in several ways to determine a candidate structure:

1) via spectral matching against spectral libraries of experimental reference spectra or *in silico* generated spectra[33,34]; 2) via machine learning-based structural predictions using experimental MS/MS-generated molecular fingerprints against structural databases[35,36]; 3) and via *de novo* structure prediction using molecular structure fingerprint prediction combined with neural networks[37].

Non-targeted LC-MS/MS metabolomics is a powerful and versatile research approach that enables high-throughput analysis and simultaneous detection of many small molecules, making it an excellent method for gaining insights into biological systems (For more information on Experimental Design and LC-MS/MS Data Acquisition, refer to **Box 1**). However, mining the vast amount of data created by non-targeted metabolomics experiments remains a challenging task despite a range of available resources that guide in the qualitative and quantitative aspects on non-targeted metabolomics. Qualitative data exploration has been democratized by platforms such as GNPS[38], by providing MS reference libraries, data analysis workflows, and compute resources for the community. Molecular networking (MN) is GNPS' core concept and is based on the comparison of all MS/MS spectra within a dataset by modification-aware similarity metrics, which network features by their similar fragmentation patterns that are often reflective of structural similarity. FBMN[6] and Ion Identity Molecular Networking (IIMN)[39] add feature detection, improving the (semi) quantitative quality within MN results. FBMN builds upon the classical MN by harnessing both MS1 information, such as isotope patterns and retention time, and ion mobility separation when used. FBMN can distinguish isomers with similar MS/MS spectra, which might remain obscured in classical MN, through chromatographic or ion mobility separation.

IIMN enhances MS/MS-based spectral networks by adding connectivity based on the MS1 feature shape correlation. It efficiently tackles the issue of unconnected ion adducts in Molecular Networking by connecting ions from the same molecules into groups called ion identity networks (IIN). This helps remove redundancy in MS-based metabolomics.

Box 1 - Experimental Design and LC-MS/MS Data Acquisition

To obtain high-quality and representative LC-MS/MS data, proper planning of the sampling and mass spectrometry analysis is essential. While this protocol article focuses on data analysis, we stress the importance of addressing the following topics prior to collecting any samples. It is crucial for researchers new to these experiments to seek guidance from a statistician and analytical chemists to guarantee optimal experimental design, instrument performance (e.g., system suitability tests), and analysis pipeline. Before proceeding with further processing and statistical analysis, raw LC-MS/MS data should be inspected by the user[40–42]. For raw data processing, we recommend the MZmine protocol[43]. Below, we provide a short checklist for guidance:

- **Experimental design and power calculation** are crucial when determining the suitable number of samples and replicates. In non-targeted metabolomics experiments, it is often challenging to predict certain values, like the feature **coefficient of variation** and the **expected effect size**, which are crucial for estimating the required sample size and the power of the study. To navigate this, reviewing previous studies with similar biological systems and research questions can provide an approximate estimation of these values.

As a general rule of thumb, when the effect size is smaller, one might need more samples or replicates.

- **Replicates (technical and biological)** to measure instrument and biological variance.
- **System and Process Blanks** to identify and correct for contamination that may be introduced during the sample collection, preparation, or measurement process. Some common blank samples include[44]:
    - **Solvent blank:** Consists of only the solvent used to dissolve the sample. It is used to identify the contaminants present in the solvent. Also, adding this blank periodically in an analytical run reduces carryover. Blanks should be added into the same well plates or vials to cover similar contaminations.
    - **Extraction blank:** It is prepared by adding a known volume of solvent to a blank matrix such as water and extracting it the same way as a sample. This extracted blank is measured along the real sample to find the contaminants introduced during the extraction process.
- **Control samples,** e.g., negative and positive controls. Depending on the experimental design, control samples are essential and should be included in the same number as the treatment samples.
- **Quality Control (QC) samples** are needed to measure instrument performance. These can be in the form of pooled QC (for example, a combination of aliquots from each sample) or standard mixtures (such as a combination of reference standard chemical compounds or isotopically labeled compounds that can also serve as internal standards). These mixtures should span a broad chemical spectrum and cover a wide retention time range.
- **Randomization of sample injection order**. It is suggested to randomize the injection order throughout the samples. However, we recommend injecting blanks at the start of the queue to prevent carry-over, which could lead to the removal of actual features from the samples during the blank removal step. Depending on the experimental design, it might also be useful to select certain sample types with the injection order, e.g., KO (knockout) vs. WT (wild type) mutant strains or low vs. high biomass samples, to avoid carryover between them.
- **Internal Standards (IS)** can be added to every sample to track instrument performance, and if desired, quantify predefined metabolites. If no internal standards are available, “housekeeping features” such as ubiquitous contaminants or metabolites can be used to control for mass and retention time drift.

183

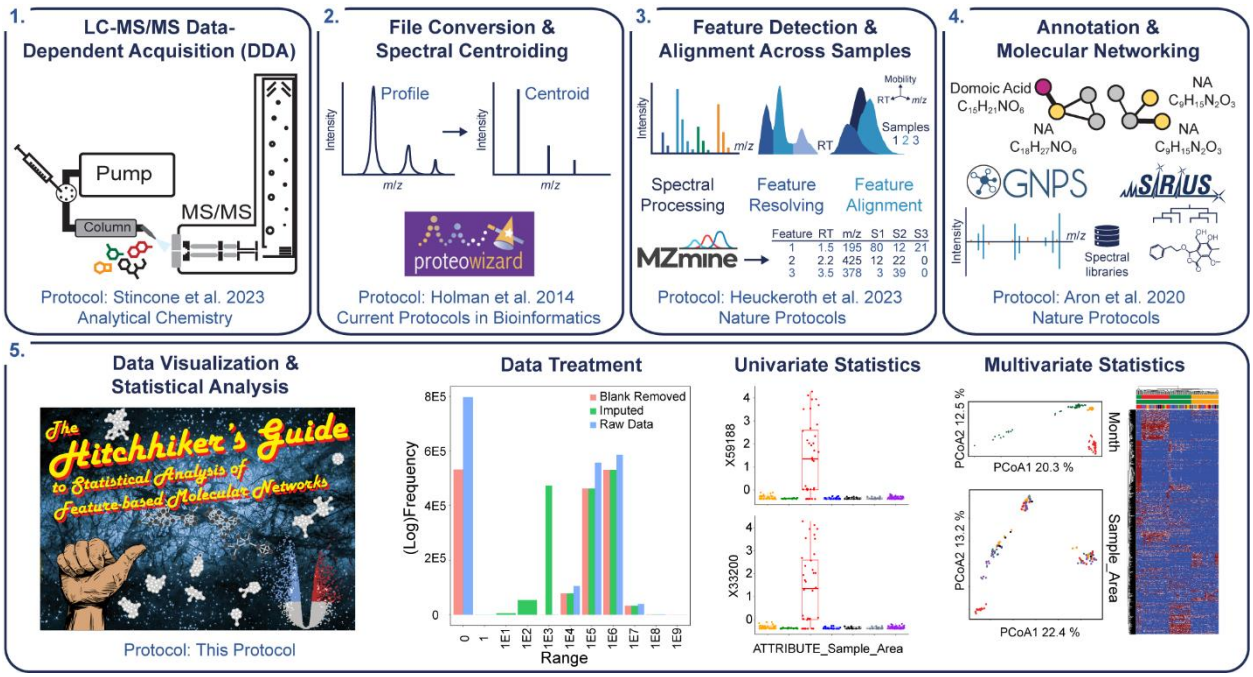


Figure 1: Flowchart of LC-MS/MS-based metabolomics experiment. 1. Data-dependent acquisition of MS/MS spectra. 2. Centroiding and file conversion. 3. Feature detection. 4. Feature annotation, network propagation and clustering. 5. Data clean-up, statistical analysis, and visualization (blank removal, imputation, normalization and scaling, followed by data visualization and statistical analysis.)

Feature-based Molecular Networking Workflow

As highlighted in **Figure 1**, non/targeted LC-MS/MS analysis workflows typically consist of data acquisition, centroiding and file conversion, feature detection, and feature annotation, including spectrum/library matching, in silico spectrum annotation and annotation propagation and clustering approaches, such as spectral networking. From there, the resulting feature tables contain all metabolites / small molecules features detected, including quantitative information (e.g., peak area) in each sample measured. This resulting feature-sample matrix is then further processed by blank removal, imputation, normalization and scaling and finally data visualization and statistical analysis.

1. File Conversion

Raw data acquisition in MS instruments entails first generating spectra in profile mode, also called continuous mode. In high-resolution instruments, each chemical entity is represented by signals of *m/z* ratios within a 5-20 ppm window, depending on the instrument's accuracy. The resulting peak profile typically approximates a Gaussian shape and is continuous. To reduce data complexity for downstream analysis, the data is simplified such that each peak in *m/z* dimension is represented by a single peak in the mass spectrum. This process is called centroiding or sometimes confusingly referred to as peak-picking, not to be confused with peak-picking in the chromatographic dimension termed "feature detection" below. Centroiding can either be

performed on-the-fly during acquisition by the vendor software or during file conversion using tools like Proteowizard's msConvert[45] or ThermoRawFileParser[46] when converting from vendor-specific formats into more accessible, community-driven formats such as mzML. When using the "vendor" option in msConvert, the centroiding algorithm provided by the instrument's vendor is used. These instrument-specific algorithms are typically more accurate than the algorithm otherwise available in msConvert and are thus highly recommended (See **Figure 1.2**).

2. Feature Detection

The process of converting raw data into a table of putative metabolic features, along with their relative abundances per sample, involves a pre-processing workflow that uses a series of algorithms. The resulting table as shown in **Table 1** is referred to as a '**feature quantification table**'. Open-source tools such as the R package XCMS[47] (often used with the package CAMERA[48] for feature grouping), MZmine 3[49], MS-DIAL[50], or OpenMS[51], in addition to vendor specific tools can be utilized for this purpose.

For the present protocol, we decided to focus on MZmine 3. Firstly, it provides an interactive and user-friendly graphical user interface (GUI) that can assist researchers without programming skills. Secondly, the direct interfacing of MZmine 3 with the downstream annotation tools is enabled by harmonized data exchange formats. Finally, the software offers a wizard for the simplified generation of data-processing workflows, which reduces the number of parameters to set and optimize for new users and experts. The process in MZmine 3 starts with importing (centroided) mzML data (described in 'File Conversion') followed by the assembly of 'mass lists' - i.e., lists of *m/z* values that exceed a user-defined intensity threshold. The workflow then progresses through three main stages: feature detection, feature alignment, and feature refinement (See **Figure 1.3**). For advanced optimization and fine-tuning, multiple tools such as NeatMS[52], MetaClean[53], and mzRAPP[54] exist to assess feature quality.

1. The feature detection phase is initiated by chromatogram building through the construction of extracted ion chromatograms (EICs) by linking MS1 signals in consecutive scans based on a maximum scan-to-scan mass deviation. This results in a list of features, each characterized by a retention time (RT) and *m/z* value. Optional smoothing in the RT dimension can be applied in the case of noisy data. The next step, 'Feature Resolving', distinguishes between overlapping and co-eluting chromatographic peaks and is used to link MS/MS spectra to their respective MS1 features. To remove redundant features originating from isotopologues of the same parent ion, the 13C isotope filter can be utilized. The 'Isotope Pattern Finder' identifies isotope signals of selected chemical elements in each feature's mass list. The steps described above are carried out for the feature list of each data file (sample) individually.
2. Next, in the feature alignment step, the individual feature lists created from multiple data files are merged by matching features across all samples based on their RT and *m/z* values. The peak alignment parameters are determined by the user and may differ depending on the particular instrument used[55].
3. Lastly, the feature refinement phase can include a gap-filling procedure that accounts for missing features in certain samples (e.g., signal below peak detection thresholds defined

in step 1). This procedure distinguishes between genuine absences and artifacts from the feature-detection process. It locates signals in the original mzML (centroided) data by re-evaluating their presence in individual samples for all the features in the merged feature list and then replaces missing values with newly detected ones based on the user-defined parameters such as *m/z* tolerance and RT tolerance. These tolerances set the window within which the algorithm finds the new feature.

Furthermore, the merged feature list can be filtered by removing duplicate features, features without a linked MS/MS spectrum, or features occurring outside a specified retention time range (e.g., during re-equilibration phases). The final step involves exporting the feature list as a feature quantification table (.csv). This table includes RT, *m/z,* and relative abundance per sample for each feature. Additionally, a text file (.mgf) is exported, describing the MS/MS and/or MS1 spectra linked to each feature. These output files provide appropriate inputs for statistical analyses and data analysis in tools such as GNPS and SIRIUS[56].

Table 1. Example Feature Quantification Table. The feature quantification table can be easily converted to a table in a text format (example of a table of features).

	m/z	RT (min)	Adduct	Charge	**Sample 1**	**Sample 2**	**…**	**Sample N**
Feature 1	97.1082	4.6	[M+H]+	+1	2.08e07	9.47e06	…	3.27e08
Feature 2	518.3032	2.0	[M+H]+2	+2	1.88e07	5.56e05	…	2.11e06
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Feature K	83.1017	1.6	[M+Na]+	+1	4.77e04	8.13e03	…	5.17e09

3. Feature Annotation, Spectral Networking and Annotation Propagation

The Metabolomics Standard Initiative (MSI) outlines four levels of metabolite identification through mass spectrometry to guide researchers in differentiating the level of identification rigor for the reported metabolites[57]. Level 1 annotations regard fully characterized compounds. To ensure accurate annotation confidence at this level, it is necessary to have at least two independent orthogonal data dimensions that match those of a pure compound analyzed in the same way (e.g., precursor *m/z,* RT, MS/MS fragmentation pattern). There should be no contradictions in any of the available data dimensions.

Level 2 confidence is assigned when data are matched against public or commercial data libraries, such as MS/MS spectral matching on GNPS. Level 3 refers to compounds whose chemical class can be putatively inferred through physicochemical features or data similarity with known compounds (e.g., by spectral similarity (networking)), or using structural prediction tools

such as CSI:FingerID or CANOPUS[35,37]. Finally, level 4 refers to unknown features that can be consistently detected (e.g., defined *m/z* value, RT and MS/MS spectrum), but could not be annotated through previous levels.

Feature annotation is essential in mass spectrometry-based metabolomics studies, especially to understand the biological significance of the detected features. Feature annotation entails several approaches, including database searches, spectral matching, and *in silico* annotation strategies. *In silico* annotation tools, such as SIRIUS, MS2Query, Network Annotation Propagation (NAP), Dereplicator, and Dereplicator+, predict metabolite identities based on spectral similarities[56] and can only lead to MSI levels 2, 3 or 4.

Another innovative method that combines feature annotation with visualization is molecular networking (MN), as shown in **Figure 1.4**. MN elucidates the structural relationships between metabolites, highlighting potential biological pathways and processes. The utility of MN spans across various fields, such as natural products, agriculture and clinical microbiology[58–60]. Using the MS-Cluster algorithm on the GNPS (https://gnps.ucsd.edu), and GNPS2 (https://gnps2.org) web platforms, MN creates a molecular network by comparing spectral similarities between each MS/MS spectra pair. With the same algorithm, GNPS allows the dereplication of MS/MS spectra by comparing them against comprehensive spectral databases, enabling feature annotations of varying reliability[7]. The .mzML or .mzXML spectra files can be analyzed on GNPS using the classical MN workflow[38].

For more precise quantitative insights, FBMN has emerged as a significant advancement by incorporating MS1 peak intensities, isotope patterns, retention times and ion mobility separation. Consequently, FBMN distinguishes between isomers with near-identical fragmentation spectra, but different retention times[6]. Unlike the classical MN, which required users to separately execute molecular networking and MS1 analysis, FBMN conveniently accepts the output of feature detection and alignment tools such as MZmine[49,61] (see 'Feature Detection' above), MS-DIAL[50], and XCMS[47], and is available on the GNPS web platform. This compatibility with other tools makes FBMN seamlessly integrated into the overall analysis pipeline.

4. Data Visualization and Statistical Analysis

The feature quantification table (see **section 'Feature Detection'** and **Table 1**), contains a list of features, such as *m/z* and RT, as well as their relative abundances per sample. This table represents the basis dataframe for statistical analyses, which can help reveal distribution patterns between sample types and determine which features are responsible for distinguishing between them. The challenge lies in prioritizing the important features in a large dataset, considering chemical and biological relevance, as well as statistical significance. An unsupervised approach for initial exploration and visualization of the data is through dimensionality reduction techniques, such as Principal Coordinates Analysis (PCoA). Ideally, such an approach will provide a 2- or 3-dimensional plot, where similar samples are grouped together, apart from dissimilar samples.

Another unsupervised statistical approach is the use of hierarchical clustering to group samples with similar relative abundance profiles of features. The results of such analysis are often visualized in combination with a heatmap. This approach displays the features within each sample colored according to their relative abundance, and groups them according to their similarity. A

dendrogram is drawn beside the heatmap to illustrate the hierarchical relationship between the samples and features. Compound class ontologies such as ClassyFire[62] or NPC[63] categorize compounds based on shared structural features or biosynthetic origins and serve as high-level annotations of the data. CANOPUS[64] predicts these compound classes from tandem mass spectrometry data without searching in structure databases. Analyzing the distribution and variety of compound classes, along with their up- and down-regulation, can yield biological insights that may not be attainable when solely considering *m/z* and retention time values.

Aim of the Protocol

The goal of this protocol is to provide an integrated pathway for downstream data clean-up and statistical analysis of FBMN results derived from non-targeted LC-MS/MS data (see **Figure 1.5**). Integrating FBMN results with statistical analyses has poses several challenges, often necessitating users to reformat, upload, and process the feature table with different external tools, in order to ultimately manually combine the outcomes. Our approach addresses this gap by offering a detailed guide and comprehensive solution to directly process and analysis the data after FBMN in one pipeline, shown in **Figure 2**. This pipeline is provided in popular scripting languages, R and Python, in conjunction with the well-known QIIME2 framework. It is available in the form of Jupyter notebooks for local use and Google Colab notebooks for cloud-based applications. Additionally, we have developed a web application with a graphical user interface (GUI), which can be accessed at https://fbmn-statsguide.gnps2.org/. The main manuscript focuses on the concepts and step-to-step guide for the R workflow, while the Supplementary Information (SI) contains step-to-step guides for the Python, QIIME2, and Webapp workflows. Though most steps are consistent across the workflows, any differences are addressed and complemented with alternative solutions in the SI. This protocol is made for both newcomers and experienced researchers in the metabolomics field:

- **For Beginners:** It introduces essential tools, resources, and workflows. The guidelines and code provided make it easier to understand common data processing and analysis steps, facilitating navigation through the complexities of the field. The provided tools utilize common programming languages (R, Python), the QIIME2 platform, and a GUI, allowing users with diverse computational backgrounds to perform data analyses.
- **For Experts:** It accelerates data analysis, ensuring faster interpretation of FBMN results.

As inputs, the protocol requires a feature table and its corresponding metadata table. Throughout its execution, users receive:

- Intermediate tables after each data cleanup step, aiding in comparison with the original feature table. Tabular outputs for clustering results from Hierarchical Cluster Analysis (HCA), list of statistically significant features as determined by ANOVA or Kruskal-Wallis tests, and Random Forest outputs indicating feature importance. Significant features refer to those that differ notably in at least one group when comparing multiple groups. Such features can be further investigated to determine if they really cause the differences we observed between groups or samples.
- Visual outputs, such as PCoA score plots, heatmaps, volcano plots for significance tests, and box plots, showcasing group differences for significant features.

This protocol helps with mapping the results of some of the statistical approaches (e.g., clustering, significant features) back to the FBMN network view. This is facilitated by importing these output feature tables, with feature IDs and the relevant information, into CytoScape in order to examine the molecular network. Moreover, as all our resources are publicly available on GitHub, users can actively raise issues or provide suggestions on GitHub.

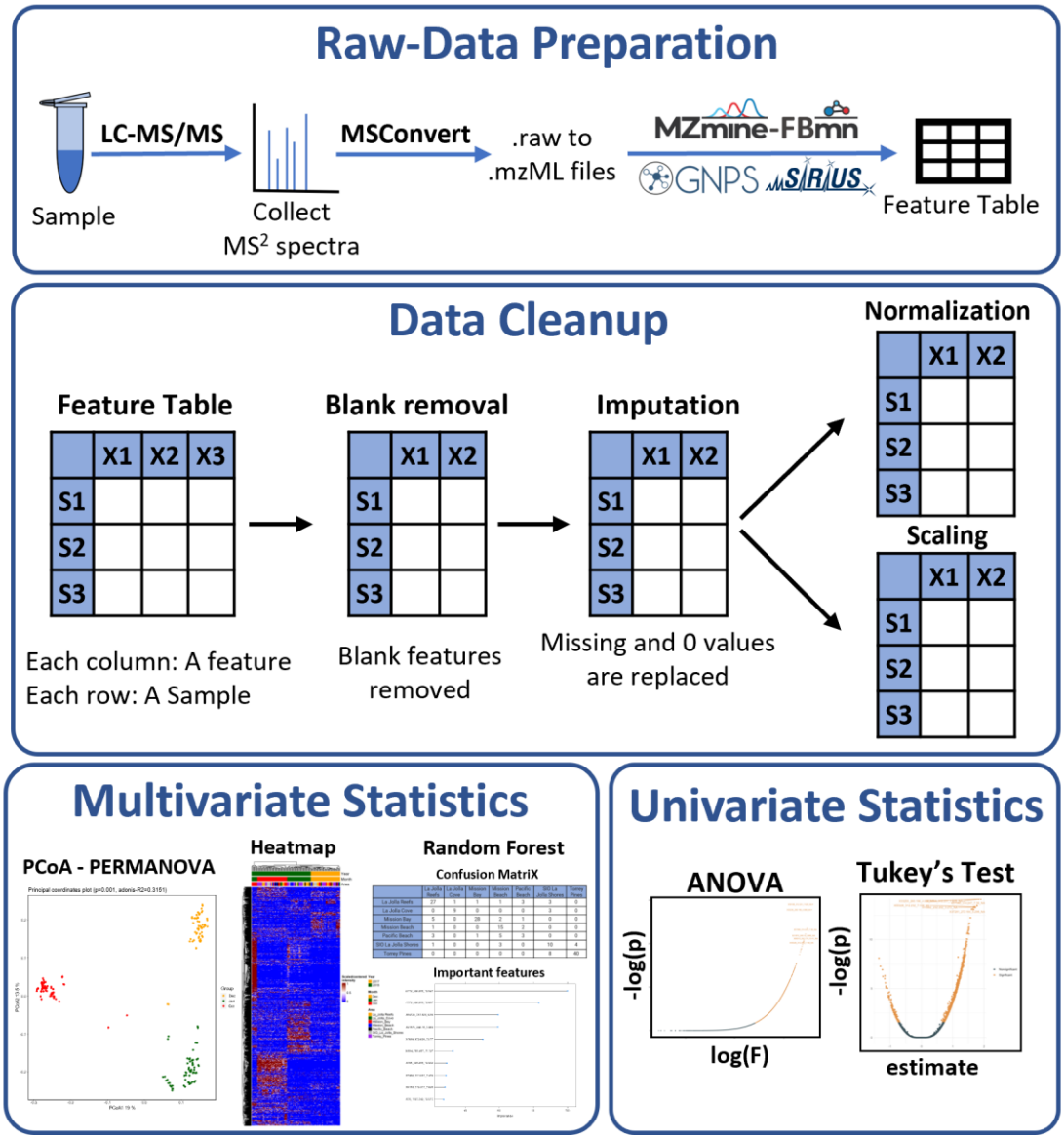


Figure 2: Overview of the Data Analysis Pipeline: Integrating four core segments, the flowchart starts with sample collection and LC-MS/MS data acquisition, transitions to raw data conversion into mzML format, and results in generating a feature quantification table under "Raw Data Preparation". This is followed by the "Data Cleanup" phase, emphasizing feature quantification table refinement, blank removal, imputation, and normalization strategies like Total-Ion-Current (TIC), Probabilistic Quotient Normalization (PQN) and scaling. Subsequently, the "Multivariate Statistics" segment showcases techniques such as PCoA plots, and heatmaps for effective data portrayal. In addition, the users are introduced to robust techniques including Random Forest classification. In the "Univariate Statistics" segment, tests such as ANOVA and Kruskal-Wallis test are discussed.

Limitations and Challenges

Our protocol for FBMN is aimed to offer advanced statistical analysis solutions for broad range of users. We thus offer the notebooks and code in different scripting languages (R, Python, and QIIME2) and platforms (Jupyter and Colab) as well as a web app to suit the specific needs and preferences within the metabolomic community.

This broad range of choices, while useful, comes with its own set of challenges. For instance, in the R Google Colab notebook, package installation can be time-consuming. Also, the inclusion of readline commands, although beneficial for customization, can appear cryptic to beginners. On the other hand, installing packages in the Python Google Colab notebook is relatively faster. There is one vital point to note regarding the 'scikit-bio' package's incompatibility with Windows. Thus, Windows OS users are advised to either use the Google Colab version or consider the Windows Subsystem for Linux (WSL) for local operations.

Furthermore, while Google Colab stands as a user-friendly platform, it is not devoid of limitations. One of the main concerns is that runtime automatically disconnects if the user leaves the Colab session inactive for 90 minutes or after 12 continuous hours of usage. This leads to the loss of the data and files they were working on from the Cloud session. Additionally, users must be aware of the 77 GB disk space limitation and ensure timely downloading of their results.

Both the R and Python notebooks comprise over 70 steps, with a significant portion dedicated to data organization. While these notebooks function smoothly with smaller datasets when run on the Cloud, their performance can lag with larger datasets (e.g., those with over 100 samples and more than 2,000 features), especially given the constraints of Google Colab. In such scenarios, local execution is advisable. For local execution, we have provided guidance on using the Anaconda Navigator, a user-friendly GUI platform, to set up Jupyter notebooks. However, MacOS users might encounter installation challenges. As an alternative, MacOS users can opt for the 'pip install' method. While numerous online resources can help with this, we have chosen not to delve into the details here. The Streamlit WebApp for the protocol, although user-friendly, has its own set of challenges. Notably, there's a data restriction of 200 MB, and larger datasets might inadvertently slow down the app or even lead to server crashes.

Lastly, the QIIME2 notebook is broadly used and applicable for both the microbiome and metabolomics communities. Our additional Jupyter notebook lets users import data directly from a GNPS job link. However, this notebook cannot be accessed in the cloud. Users need to either install QIIME2 and GNPS packages on their computer or use Docker. This might be difficult for some, but it is a good option for those familiar with QIIME2[65]. In all cases, users should always consider the size of their data, their computer's power, and their own skill level while using the protocol.

Alternative Open-Source Data Analysis Workflows and Protocols

There have been many efforts in the community to provide and teach statistics solution for non-targeted metabolomics data analysis, and multiple, scripts, web apps and software tools are available for data clean-up, statistical analysis and visualization. While we believe that such a streamlined solution for FBMN results, as described in our protocol, has not yet been provided, we would like to point out the many other tools, workflows and applications that are available.

Table 2: Overview of alternative statistics tools and scripting solutions for statistical analysis of non-targeted metabolomics data.

Tool Name	Tool Type	Availability	Raw Data Processing	Blank Removal	Matrix Transformations	Uni-Variate Statistics	Multi-variate Statistics	Export for Downstream Tools	Customizable	URL	Reference
GUI											
MetaboAnalyst	Web App (GUI)	Open Source	Y	Y	Y	Y	Y	N	N	www.metaboanalyst.ca/	**23,24**
Workflows											
Galaxy-M	Workflow	Open Source	Y	Y	Y	Y	Y	N	N	github.com/Viant-Metabolomics/Galaxy-M	**66**
Workflow4Metabolomics	Workflow	Open Source	Y	Y	Y	Y	Y	N	N	github.com/workflow4metabolomics	**67**
UmetaFlow	Workflow	Open Source	Y	Y	Y	Y	Y	N	N	github.com/biosustain/snakemake_UmetaFlow	**68**
Chemometrics Tutorials	Workflow / Tutorial	Open Source	N	N	Y	Y	Y	N	Y	github.com/Gscorreia89/chemometrics-tutorials	
QIIME2 metabolomics plugin	Language	Open Source	N	N	N	Y	Y	N	N	library.QIIME2.org/plugins/q2-metabolomics/10/	**65**
R Libraries											
mixOmics	Package	Open Source	N	N	Y	Y	Y	Y	Y	mixomics.qfab.org	**69**
MetaboanalystR	Package	Open Source	Y	Y	Y	Y	Y	Y	Y	www.metaboanalyst.ca/docs/RTutorial.xhtml	**70**
omu	Package	Open Source	N	N	Y	Y	Y	Y	Y	cran.r-project.org/web/packages/omu/vignettes/Omu_vignette.html	**71**
metabolomicsR	Package	Open Source	N	N	Y	Y	Y	Y	Y	cran.r-project.org/web/packages/metabolomicsR/index.html	**72**
MAIT	Package	Open Source	N	N	Y	Y	Y	Y	Y	www.bioconductor.org/packages/release/bioc/html/MAIT.html	**73**
ropls	Package	Open Source	N	N	Y	Y	Y	Y	Y	bioconductor.org/packages/release/bioc/html/ropls.html	
MSStats	Package	Open Source	N	Y	Y	Y	Y	Y	Y	github.com/Vitek-Lab/MSstats	**74**
Python Libraries											
TidyMS	Package	Open Source	Y	Y	Y	N	N	Y	Y	github.com/griquelme/tidyms	**75**

We summarized those that, in our opinion, are the most commonly used software tools in **Table 2.** This table provides an overview of their functions, purpose, tool type, and when applicable, references to related protocols and guidelines. We also indicated where in the data processing pipeline these tools have application by indicating yes ("Y") or no ("N") in columns related to raw data processing (generation of feature quantification table, see **section 'Feature Detection'**), data clean-up steps (involving quality control, missing value imputation, normalization, scaling, and transformation), and multivariate and univariate analyses.

We do note that this table is by no means exhaustive. All of these options are workflow dependent and vary based on factors such as the structure of the acquired feature quantification table and the chosen data analysis techniques[76], and typically require specific file and table formats.

Expertise Needed to Implement the Protocol

We aimed to make this protocol accessible to a broad range of researchers, from absolute beginners to experts. As we provide different options of executing the code (Web App, Colab and Jupyter notebooks), the protocol should be useful for users both new to metabolomics data analysis, who want to perform a fixed set of processing and statistical analysis, as well as users that require customizable options and need to analyze large datasets. To guide readers through the different options and help to choose which option is most suitable, we generated a decision tree displayed in **Figure 3**. At a minimum, we recommend to have some general background in statistics and a basic understanding of LC-MS/MS data structure, as well as knowledge about the biological system and the experimental design of the dataset which should be analyzed.

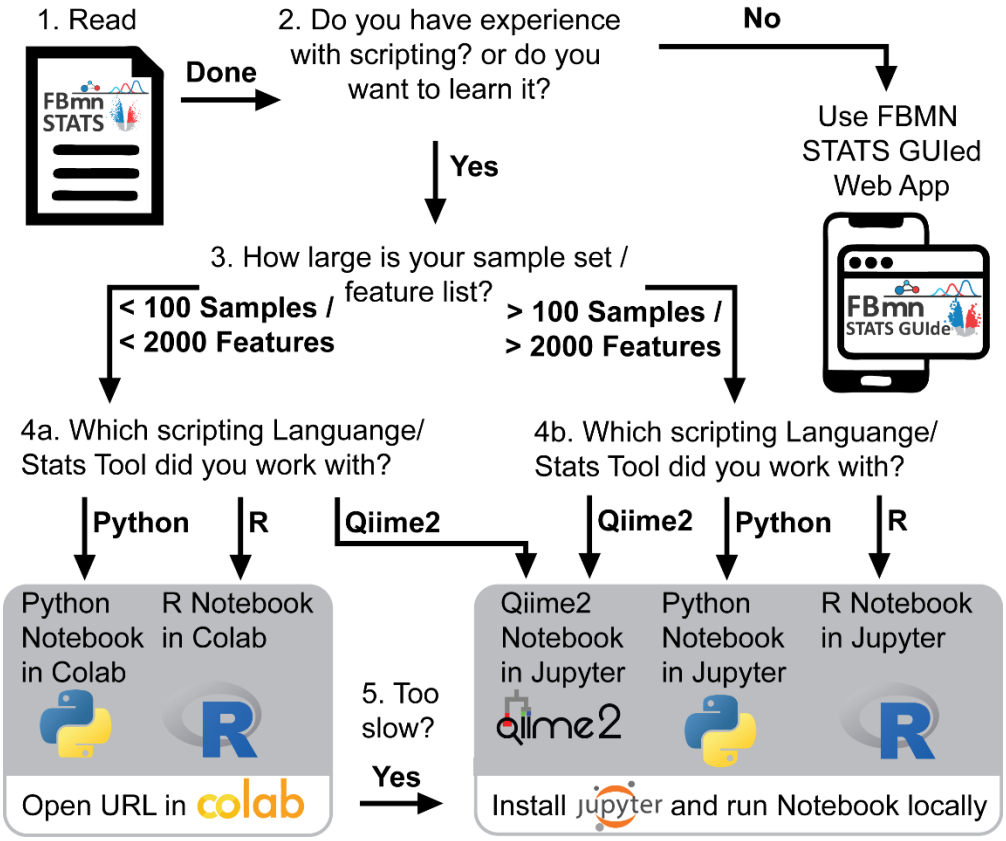


Figure 3: Decision tree to guide choosing which notebook/app to use.

2. Materials

Software Used

- Google Colab (Optional, cloud)
- Local installation of Jupyter Notebook (Optional)
- Streamlit Web App (Optional)
- QIIME2 (Optional)

Note: The pipeline can be accessed through Google Colab (which requires no local software installation), a Web App with Graphical User Interface (GUI), or through a local installation of Jupyter Notebook, which may be preferred for larger datasets. A decision tree for which method to choose is provided in **Figure 3**. As a default for beginners, we recommend using the Colab Notebook with R code. In addition to the R code, Python and QIIME2 versions are also available in our GitHub Repository (https://github.com/Functional-Metabolomics-Lab/FBMN-STATS/tree/main/data). Additional information regarding files other than R code available are provided in our supplementary information. A Web App version of the protocol (https://fbmn-statsguide.gnps2.org/) is available for those who prefer a more visual interface.

Required Files:

- Feature quantification table (.csv)
- Metadata (.txt)

The feature quantification table (.csv), a characteristic product of LC-MS/MS metabolomics studies, encompasses all mass spectral features (integrated peak areas) and their relative intensities across diverse samples. As mentioned earlier, we used MZmine 3 to obtain the feature quantification table in .csv file format.

The metadata is a .txt file that can be created in a word editor or spreadsheet programs such as excel or google sheets. The metadata table needs to be created by the user, providing additional context for the measured samples, such as sample type, species, tissue type, and collection time point. For the datasets to be fully considered for public meta-analysis, we suggest using a standardized metadata format with controlled vocabulary. We recommend adhering to standardized metadata practices and protocols, and for guidance users can refer to the ReDU metadata template (https://mwang87.github.io/ReDU-MS2-Documentation/HowtoContribute/).

The metadata format in this protocol should be compatible with GNPS workflows (https://ccms-ucsd.github.io/GNPSDocumentation/metadata/). The first column in the metadata, labeled 'filename', should match the exact filenames as reported in the feature quantification table. Following this, one should include additional columns to the metadata that begin with 'ATTRIBUTE_' (e.g., ATTRIBUTE_groups, ATTRIBUTE_timepoint).

In our example metadata, columns like ATTRIBUTE_Replicate, ATTRIBUTE_Sample_Type, ATTRIBUTE_Batch, ATTRIBUTE_Month, and ATTRIBUTE_Year all contain group-based information. This type of grouping will assist in selecting different categories for statistical analysis throughout this guide. You can also include columns with continuous numerical data, such as

ATTRIBUTE_Injection_order or ATTRIBUTE_timepoint. To ensure statistical power, it is essential to use replicates (we suggest at least three) for each sample type within the experimental design. See **Table 3** for an illustration of the metadata structure.

Table 3: Sample metadata layout.

The first column, 'filename', lists the filenames along with their specific extensions (preferably 'mzML' or the older 'mzXML'), exactly matching the column names in the feature quantification table. Two example "ATTRIBUTE_" columns are also included: "ATTRIBUTE_groups", which showcases sample categorical data (i.e., different sample types such as Control, Sample, and Blanks), and "ATTRIBUTE_timepoint", which is an example for numerical data.

filename	**ATTRIBUTE_groups**	**ATTRIBUTE_timepoint**
control_rep1.mzML	Control	1
⋮	⋮	⋮
Sample_type1_rep1.mzML	Sample_type1	4
Sample_type1_rep2.mzML	Sample_type1	4
⋮	⋮	⋮
blank.mzML	Blank	NA

Additional Input Files

Besides the feature quantification table and metadata, the R and Python notebooks can also integrate molecular annotation files (either in .txt or .tsv format). These include SIRIUS, CANOPUS, and GNPS annotations, which enrich our understanding of each feature during analysis. While the inclusion of SIRIUS and CANOPUS files is optional, they can provide valuable insights.

GNPS annotations can be obtained from the Feature-Based Molecular Networking (FBMN) analysis. The process requires MS/MS fragmentation patterns in the ".mgf" format, a feature quantification table, both obtained with e.g., MZmine 3 (see **section 'Feature Detection'**), and a metadata file. The .mgf file carries spectral information about specific MS/MS scans designated for each feature and feature IDs match with feature IDs in the feature quantification table. All of these files need to be uploaded to the GNPS platform.

The metabolite annotation requires a user-defined mass tolerance. Subsequently, MS/MS patterns are matched against the GNPS database using a modified cosine similarity[77], resulting in a molecular network that allows for the identification of compound names for all library hit features. The output of the FBMN job associated with the example data of this protocol is publicly available

https://gnps.ucsd.edu/ProteoSAFe/status.jsp?task=b661d12ba88745639664988329c1363e and

can be downloaded using the 'Download Cytoscape Data' option. The FBMN job's .graphml file, found under the folder "gnps_molecular_network_graphml", can be used to visualize the molecular network in Cytoscape software. The respective annotated files are located in the "**DB_result**" and "**DB_analog_result**" sub-folders (assuming an analog search is performed), with the former offering level 2 and the latter providing level 3 (molecular class) annotations. The analog search identifies structurally related molecules within the molecular network by applying a score threshold, such as a minimum cosine score that MS/MS spectra must achieve to be considered an annotation during spectral matching with MS/MS spectral libraries. An upper limit can be established for the mass shift between the library and potential analogs (e.g., 100 Da), thus expanding the scope of annotation.

SIRIUS[56] can predict molecular formulas, as well as structures through structure database matching using CSI:FingerID[35,78]. Furthermore, the integrated CANOPUS[64] module provides ClassyFire based chemical class predictions. As for GNPS, the required input is a .mgf file associated with the MZmine feature quantification table with matching feature IDs across both files. However the .mgf file exported for SIRIUS through MZmine 3 differs from the .mgf exported for GNPS in that it contains isotopic MS1 patterns for accurate molecular formula prediction.

All example input files to follow this protocol can be retrieved from the Functional Metabolomics Lab GitHub Repository (https://github.com/Functional-Metabolomics-Lab/FBMN-STATS/tree/main/data). Furthermore, users have the convenience of directly uploading all input files by simply entering the task ID from their FBMN job on GNPS.

Example Dataset

The example dataset is part of a previously published study[14], aimed to elucidate the effects of urbanization on organic matter chemotypes in coastal environments after a major rainfall event. Seawater samples were collected from 30 locations over seven areas along the San Diego, California coastline: Torrey Pines, SIO La Jolla Shores (Scripps Institution of Oceanography at La Jolla Shores), La Jolla Cove, La Jolla Reefs, Pacific Beach, Mission Beach, Mission Bay, capturing both pre- (Dec 2017) and post-rainfall (Jan 2018) conditions. In our analysis, we included supplementary data from October 2018, collected from the same sites (no-rain period), for our pipeline evaluation. The dataset consisted of 180 samples from the three sample times (Dec 2017, Jan 2018, Oct 2018) and 2 PPL process blanks at each of the sample times. The datasets can be found in the MassIVE repository: MSV000082312 and MSV000085786 https://massive.ucsd.edu/ProteoSAFe/dataset.jsp?task=8a8139d9248b43e0b0fda17495387756 https://massive.ucsd.edu/ProteoSAFe/dataset.jsp?task=c8411b76f30a4f4ca5d3e42ec13998dc

Note: Seawater samples collected during October 2018 were not available in the original article yet. The .mzML files were preprocessed using MZmine 3 (https://mzmine.github.io/) and the feature-based molecular networking workflow in GNPS (https://gnps.ucsd.edu/ProteoSAFe/status.jsp?task=b661d12ba88745639664988329c1363e).

3. Procedure

This protocol primarily focuses on the R workflow, given its broad adoption in metabolomics data analysis and the extensive libraries it offers for this purpose. However, recognizing the diversity of our audience and the growing popularity of other platforms, we've also developed workflows in Python and QIIME2 as well as a web application. Please refer to the SI document for the Python/QIIME2 notebook or web app workflow. In the following sections, instead of step-by-step instructions, we highlight the key concepts to avoid repetition from the notebook. Code blocks are included to illustrate the main algorithms and functions.

▲CRITICAL: We recommend initially executing the notebook using the provided example dataset. Once familiar, proceed with your own data. This approach ensures a smooth transition from learning to applying the workflow.

General Instructions for Navigating the Notebook:

- **Text in Red**: These sections indicate critical information or cells that require user input within the notebook. They serve as instructions for adapting the notebook to different datasets without the need to modify the code. Further details are provided within the notebook.
- **User Prompt Guidance:** When you encounter code cells with red highlights, simply execute them without changing their contents. For instance, you may come across lines such as

  ```
  Directory <- normalizePath(readline("Enter the folder path in the pop-up box: "),"/",mustWork=FALSE)
  ```

  To provide input, a pop-up box will appear in the output section. Make sure to enter your answers in the pop-up box instead of entering directly within the code. After entering your input, remember to press 'Enter' to proceed to the next step.
  Using these prompt boxes ensures that user input is seamlessly integrated into the following operations. The position of these prompt boxes might differ depending on your system as they could appear directly below the active cell, at the notebook's top, or even towards the upper section of your screen.
- **Text in Green**: This indicates that the following cell in the notebook contains function definitions and will not display any visible outputs. Even though the underlying code in these cells may seem complex, its purpose is to make repetitive tasks more efficient. Readers who come across these green-highlighted cells do not need to understand the complexities of the code.
- **Using the '#' Operator**: Lines in the code cells that start with '#' are comments explaining the code's function or purpose. These comments are "commented out" and will not be executed. To run a commented-out code, remove the '#' symbol and run the cell again.

3.1. Preliminary Setup for the Notebook

We recommend beginners to use Google Colab for the R notebook due to its hassle-free setup as it requires no installations, making it accessible for those unfamiliar with the setup process. However, for regular analysis, local execution in Jupyter on a contemporary desktop computer

(E.g., Intel i7, 16 core, 64 GB RAM) is typically faster and more efficient. The reported processing times here are based on our example data set on the Colab platform. The durations other than for package installation are estimated from a beginner's viewpoint, reflecting the time typically required for someone new to complete the analysis. To easily install and run Jupyter Notebook in R, consider using Anaconda Navigator, following the instructions provided in the accompanying document (https://github.com/Functional-Metabolomics-Lab/FBMN-STATS/blob/main/Anaconda_Rkernel_installation_JupyterNotebook_JupyterLab.pdf).

▲**CRITICAL**: To ensure proper execution and chronological order, please run the notebook cell-by-cell instead of running multiple cells simultaneously. The numbers assigned to each cell will help you navigate and determine if the cells have been executed correctly and in chronological order.

3.1.1. Package Installation ● Timing 15 mins

Step 1: Package Installation

- The notebook utilizes R version 4.1.3 (2022-03-10)
- Begin by installing and loading the necessary R packages using the `p_load()` function from the 'pacman' (v0.5.1) package[79]. This function checks if a package is installed, if not, it installs the package from CRAN or other repositories in the pacman repository list and loads the package. It is a more efficient alternative to using `install.packages()` and `library()` functions individually for each package.
- **Required Packages:** The following R packages are essential for this protocol:
    - Data Cleanup: tidyverse[80] (v2.0.0), IRdisplay[81] (v1.1), KODAMA[82] (v2.4)
    - Multivariate Statistics: factoextra[83] (v1.0.7), vegan[84] (v2.6-4), ComplexHeatmap[85] (v2.10.0), dendextend[86] (v1.17.1), NbClust[87] (v3.0.1), rfPermute[88] (v2.5.1).
    - Univariate Statistics: FSA[89] (v0.9.4), matrixStats[90] (v0.63.0).
    - Visualization and Plotting: ggsci[91] (v3.0.0), cowplot[92] (v1.1.1), svglite[93] (v2.1.1).
- Packages are installed just before their respective sections to reduce installation time. However, please note that the packages installed initially in one section can be used for the later sections as well. For example, tidyverse (v2.0.0) can be used throughout the notebook, not just for data cleanup.

Step 2: Set Working Directory

(User Input Required)

- Set a folder as the working directory. This is where you access input files and save output files. Make sure to include all necessary input files in this folder.

- In Google Colab, click on the three dots in the upper left corner to see the notebook contents. Click on the folder icon and create a new folder by right-clicking in the empty space and selecting 'new folder'.
- When you run the following cell in the notebook to set a working directory, a pop-up box will display as shown in **Figure 4**. Insert the path of the folder containing your input files and press 'Enter'.
- **(For Local Environment)** If you're running the notebook in your local environment, you can directly specify the local path of your folder to set it as your working directory. For example, if your folder is located at D:\User\Project\Test_Data, simply input this path when prompted and press 'Enter'.

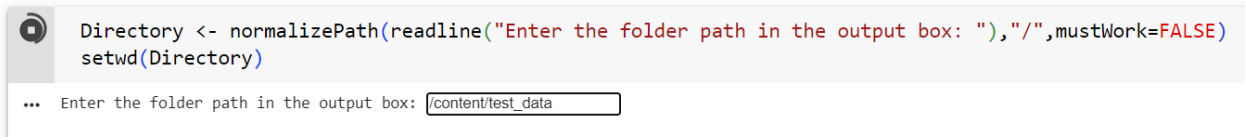


Figure 4: Screenshot of the code cell from R Google Colab Notebook to set the working directory. The image displays the code cell targeting the '/content/test_data' directory. This user-created directory holds the input files for the data analysis. Note the stop symbol with the surrounding loading circle, indicating the cell awaits user input. To proceed to the next cell, provide the input (e.g., /content/test_data) and press enter.

3.1.2. Data Import

This section guides users through the process of importing necessary data files for the Notebook. Various methods are outlined, catering to different data sourcing preferences.

Step 3: Uploading Files to Google Colab
(User Input Required)

Right-click on the folder you created in the Google Colab workspace and select 'upload' to transfer the required files from your local machine to the cloud session. If you do not want to use files from your local machine, you can skip this step and proceed directly to step 4.2 ('Loading Files from URL') or 4.3 ('Loading Files from GNPS').

Step 4: Select a Data Loading Method (Choose One Option from Steps 4.1 to 4.3)
The user can choose from the following options provided in steps 4.1 to 4.3 to import their data.

Step 4.1: Loading Files from the folder
(User Input Required)

a)

A data.frame: 5 × 2

INDEX	FILENAMES
<int>	<chr>
1	20221102_SD_BeachSurvey_batchFile.xml
2	20221125_Metadata_SD_Beaches_with_injection_order.txt
3	GNPS_analog_result_FBMN.tsv
4	GNPS_result_FBMN.tsv
5	SD_BeachSurvey_GapFilled_quant.csv

b)

```
input_str <- readline('Enter the index number of the feature table and metadata separated by commas: ')
```

```
Enter the index number of the feature table and metadata separated by commas:
5,2
```

Figure 5: Screenshots illustrating loading input files from a folder: a) Table showing all the files in the working folder, where the first column, labeled **"INDEX"**, denotes the serial or **index number** of the files. b) Shows the user input interface. Upon executing the code cell, the user is prompted to enter the index numbers for the feature table and metadata. In this example, "5" and "2" are entered, referencing the files indicated in (a).

In this step, you will begin by viewing a table displaying a list of files in your working folder (e.g., uploaded in the previous step) as shown in **Figure 5a**. Each file will have an index number associated with it. Your task will be to import three tables by specifying the index number associated to each: the feature quantification table (`ft`), the metadata table (`md`), and optionally, annotation tables (`an`). For an example, see **Figure 5b**. To guide you through this process, there are three code blocks that require user input.

1. **Feature Quantification Table and Metadata Import:** The first code block will prompt you to enter the index numbers associated with the feature quantification table and metadata, separated by commas. Simply input the corresponding index number assigned to each of these files.
2. **Annotation Tables Import:** The second code block will request the index numbers associated with the annotation tables. Specifically, you will be asked to enter the index numbers of the GNPS library annotation file and the analog annotation files. If you have not performed an analog search for FBMN, only provide the index number of the GNPS library annotation file.
3. **SIRIUS Annotation File Import (Optional):** The third code block requests you to input the index number of a SIRIUS annotation. This file will be used to merge all annotations (e.g., GNPS library, analog hits, SIRIUS) into a single master table for easier data exploration later on. It is worth noting that this protocol does not specifically focus on SIRIUS annotations for analysis. The inclusion of SIRIUS annotations is solely for the convenience of consolidating all annotations in one place for the user.

By following these prompts, one can successfully load the essential tables required for the subsequent analysis. Make sure to carefully input the correct index numbers.

Step 4.2: Loading Files from URL

(User Input Required)

- We also provide an example of retrieving data from a URL (for example, the feature quantification table can be obtained from https://raw.githubusercontent.com/Functional-Metabolomics-Lab/FBMN-STATS/main/data/SD_BeachSurvey_GapFilled_quant.csv)
- We access the feature quantification table, metadata, and analog result files directly from our Functional Metabolomics GitHub page.
- If you are using your own dataset (or the test dataset) in Google Colab, you can get the file URL by uploading the input files to the Colab workspace, right-clicking on the file, selecting "Copy path", and then replacing the URL in the relevant cell.

Step 4.3: Loading Files from GNPS

(User Input Required)

- In this step, you can load files directly from the repositories MassIVE or GNPS. If you have performed FBMN on your feature quantification table, you can access the required files by providing the task ID.
- To locate the task ID of your FBMN job within your GNPS account, navigate to the 'Jobs' section. Here, the 'unique ID' for each job is listed in the 'Description' column.
- When you run the relevant cell in the notebook, you will be prompted to enter the task ID within the notebook. Given the task ID, the notebook will retrieve the necessary files for further analysis.

▲**CRITICAL:** Make sure your metadata has the necessary attribute columns to describe the data (at least one, e.g., ATTRIBUTE_SampleType). If your FBMN metadata is insufficient, you might need to load additional metadata from a local folder for downstream statistical analysis, adding an additional step in the workflow.

Step 5: Exploring the Imported Files

Use the `head()` and `dim()` functions to get an initial view of your imported data files.

- The `head(ft)` function displays the first six rows of the feature table by default, giving you a quick look at your data's structure.
- The `dim(ft)` function reveals the dimensions of your feature quantification table, i.e., the number of rows and columns.

▲**CRITICAL:** If you encounter an error while executing certain code cells, it is good practice to verify the correctness of your data tables using the `head()` and `dim()` functions.

We also provide a special summary function `InsideLevels(md)` to explore the metadata, which returns a summary table with columns for INDEX, ATTRIBUTES, LEVELS, COUNT, and ATTRIBUTE_CLASS.

1. **INDEX**: Row number in the summary table
2. **ATTRIBUTES**: Column name of the attribute, e.g., ATTRIBUTE_Sample_Type
3. **LEVELS**: Unique groups within the attribute column, e.g., Blanks, Sample
4. **COUNT**: Number of files for each category, e.g., 6, 180 indicating 6 files for "Blank" sample type and 180 for "Sample" sample type.
5. **ATTRIBUTE_CLASS**: Data type of the attribute. Useful for spotting cases where a numeric attribute like ATTRIBUTE_minutes is classified as 'character'.

3.1.3. Merging Annotations with Feature Quantification Table

This section involves integrating various annotations, such as SIRIUS and GNPS annotations, into our feature quantification table. This process is vital as it helps identify the metabolites corresponding to the features in our feature quantification table, aiding in the interpretation of our metabolomics data.

Step 6: Identifying Appropriate Columns for Merging

Depending on the type of annotation to be merged, the feature quantification table's unique 'row ID' column is matched with the corresponding column in the annotation file:

- **GNPS Annotations:** The '`row ID`' is matched with the '`#Scan#`' column in the GNPS annotation file. The '`Compound_Name`' column contains the annotation information.
- **GNPS Analog Annotations:** Similar to GNPS annotations, the '`row ID`' is matched with the '`#Scan#`' column in the GNPS analog annotation file. The '`Compound_Name`' column contains the annotation information.
- **SIRIUS Summary Files:** The '`row ID`' is matched with the '`id`' column in the SIRIUS summary file. A typical feature ID in the '`id`' column might look like this: "3_ProjectName_Mzmine 3_SIRIUS_1_16", where the last string (16) represents the row ID.

Step 7: Ensuring Data Compatibility

Before merging, we ensure that the classes (or data type) of the columns meant to be merged are the same. Then, we can combine feature and annotation data based on the appropriate matching columns. Any mismatch, such as one column being of character type while the other one is numeric, can cause merge errors, even if the values within the columns are identical.

Step 8: Merging Annotations

- Rename the column names of analog annotation dataframe `` `an_analog` `` with an 'Analog_' prefix and merge the modified `` `an_analog` `` dataframe with `` `an_gnps` `` based on #Scan#.
- For each unique '#Scan#', consolidate multiple compound names into a single row. If both the GNPS compound names (actual and library hits) for a particular '#Scan#' are identical,

keep one; otherwise, combine them using a ";" separator. The output is `an_final_single`.

- Merge `an_final_single` with the feature quantification table (`ft`) using '#Scan#' and 'row ID' as matching columns respectively. Keep all rows from the feature quantification table.

Additional steps:

Incorporating Additional Annotations (optional)

- If SIRIUS annotations are available, follow these additional steps: Extract 'row ID' from the 'id' column in the SIRIUS dataframe, rename the columns with a 'SIRIUS_' prefix, and merge the modified SIRIUS dataframe with `an_final` data frame based on 'row ID'.
- For simplicity, we have shown here how to merge SIRIUS summary files. This process can be similarly adapted for CANOPUS summary files.

Exporting the Merged Annotations

- Finally, write the merged annotation table to a CSV file for further data exploration and downstream analyses.

3.1.4. Ensuring Metadata and Feature Quantification Table Compatibility for Downstream Analysis

This section streamlines the metadata and feature quantification tables, ensuring they align perfectly for subsequent steps in the protocol and remove discrepancies between them. By following the outlined steps, we achieve harmonized data structures. A final verification confirms that all files in the feature table are mirrored in the metadata, and vice versa. Upon successful validation, the tables are set for the next section of analysis. If there is a mismatch, often due to naming inconsistencies or missing files, the user needs to rectify these issues before moving forward. As a user, you are not required to modify any of the code within this section. Simply execute each cell in turn.

Step 9: Creating Backup Files

The feature quantification table (`ft`) and metadata (`md`) files are stored under different names (`new_ft`, `new_md`) to preserve the original versions.

Step 10: Cleaning up the Feature Quantification Table

- Clean the feature quantification table by removing 'peak area' extensions from the column names, a default format included in MZmine-derived feature quantification tables.
- Check and remove any columns containing only NA values present in the feature and metadata tables.
- Check and remove any rows and columns containing only empty strings in the metadata table

Step 11: Updating the Row Names of the Feature Quantification Table

- In this step, we reformat the row names to consolidate essential information about each feature. By doing this, we can retain only the numeric data in the feature quantification table and remove all other columns.
- The row names are constructed by concatenating the Feature ID, *m/z*, RT, and GNPS annotations into a single string, in the following format: `XFeatureID_*m/z*_RT_GNPS_annotations`.
- An example row name is `**X92649_226.951_14.813_NA;TRYPTOPHAN**`. Here, "NA" indicates that there was no direct library hit for this feature. However, the analog annotation suggests it could be tryptophan.
- In the R environment, a dataframe's row names must be characters or strings. Thus, we add the 'X' character prefix to the numeric Feature ID to ensure compatibility.

Step 12: Selecting Relevant Columns

(User input - Optional)

- Retain only '.mzML' (or '.mzXML') file-relevant columns and remove extraneous information, such as additional columns added due to IIMN. Here, the features are represented as rows in the feature quantification table.
- Only when the file extensions '.mzML' or '.mzXML' are not available, the user is prompted to enter their respective file extension.
- This step ensures that the feature quantification table contains only the intensity values of the features, which is crucial for subsequent calculations. The modified row names provide basic feature information, and for a more detailed understanding, you can refer back to the original feature quantification table.

Step 13: Verifying File Consistency

The metadata and feature quantification tables are arranged in the same order of '.mzML' (or '.mzXML') file names. We then verify consistency between the feature and metadata tables by using the `identical(new_md$filename, colnames(new_ft))` command.

- If the result is TRUE, proceed to data cleanup.
- If FALSE, there might be missing files or discrepancies in file naming. Check the corresponding column names in the feature quantification table for potential errors like spelling mistakes or case-sensitive issues, and re-upload the correct files. Re-run all the above steps once corrected.

3.2. Data Cleaning: ● Timing 20-30 mins

Following the LC-MS/MS data pre-processing with MZmine, we perform the post-processing of the data (also known as data pretreatment or data clean-up) as the first crucial step in our workflow. While the 'preliminary setup for the notebook' section prepares the feature and metadata tables for analysis, actual modifications to the data commence from this section.

Step 14: Transposing the Feature Quantification Table

- As a first step, we transpose the feature quantification table. The result is a table (`ft_t`) where the row names represent the sample names, and the column names consist of concatenated feature information.
- Then, we merge this transposed feature quantification table (`ft_t`) with the metadata (`new_md`), using the sample names as the common link. This merged table, referred to as `ft_merged`, consolidates all necessary information in a single structure.
- The `ft_merged` table can also be exported to a CSV file for future use, such as batch correction or other specialized analyses.

3.2.1. Batch Correction (Optional)

Batch effects are systematic differences in sample measurements when samples are run as multiple batches or groups. In most cases, when the sample sizes exceed the measurement assay, it is often necessary to measure the samples in multiple batches. This might lead to varying mass spectra among the samples within a batch and among different batches[94]. Several factors could contribute to these batch effects such as variability in instrument conditions, RT shifts, and gradual contamination of LC columns when measuring multiple samples over a long period. These are often unavoidable issues, hence it is common to treat these effects post-sample measurement[95].

To correct these unwanted variations, we first need to identify their presence, remove or adjust the variations for further statistical analysis and assess the performance of our method[96]. The most common method for visualizing or identifying the presence of batch effects is through a simple Principal Component Analysis (PCA), guided PCA[97] or Principal Coordinates Analysis (PCoA). In the PCA/PCoA scores plot, it is generally expected that all the QCs cluster together indicating little analytical variation in the data. When the inter-batch variation gets higher, the inter-QC distances in the PCA/PCoA plot will also increase[98]. To visually assess this using the notebook, follow these steps:

1. Execute Step 25 to install necessary packages for multivariate analysis.
2. Run Step 32 and Step 33 to visualize the PCoA using the custom-made `plotPCoA()` function. Detailed usage instructions are provided in the respective steps.

Assuming your sample type information (description of which samples are pooled QCs, blanks, samples etc.) is located in the 'ATTRIBUTE_Sample_Type' column of the metadata, the function can be invoked as follows:

```
plotPCoA(
    ft = ft_t,
    md = new_md,
    distmetric = "euclidean",
    category_permanova = "ATTRIBUTE_Sample_Type",
    pcoa_category_type = 'categorical',
    category_pcoa_colors = "ATTRIBUTE_Sample_Type")
```

We have deferred this visualization to a later section, after data cleanup. As we delve deeper into multivariate analyses after data cleanup, this approach avoids redundancy and ensures users can maximize the utility of this protocol.

Another method is to use Analysis of variance (ANOVA) by comparing the QC mean of different batches for statistically significant differences[99]. Once the presence of a notable batch effect is confirmed, multiple approaches can be used to correct the effects, including

- **Normalization** methods such as Metabodrift[100], ComBat[101]
- **Transformation** method: waveICA[102] (wavelet transformation coupled to ICA)
- **Regression-based** approaches such as linear least-square (LS) method[103], QC-based robust LOESS correction[40], QC-support vector regression[104] (QC-SVR)
- **ML-based** methods such as random forest-based QC-RFSC correction[105], deep learning model: NormAE (Normalization Autoencoder) algorithm[106], Regularized Adversarial Learning Preserving Similarity[107] (RALPS).

Each method has its strengths and limitations. When there are no QCs included in the study, normalization can be used instead to attempt to reduce most of the unwanted variations[108], at the risk of removing true biological variation. For the sake of simplicity and to cater primarily to beginners, this protocol does not elaborate on batch correction. However, for those interested in exploring batch correction in depth, we have prepared a supplementary R notebook available on our GitHub repository (https://github.com/Functional-Metabolomics-Lab/FBMN-STATS/blob/main/R/Additional_Notebooks/Batch_Correction.ipynb). In this notebook, we execute inter-batch correction similar to the method described by Qin Liu *et al*[94]. The procedure involves calculating the mean of each feature across all batches, then calculating the batch-specific feature mean, and subsequently adjusting feature intensities within each batch relative to the batch-specific and overall means. For intra-batch adjustments, the notebook illustrates the QC-based LOESS correction method, with a prerequisite that each batch should start and end with a pooled QC injection.

3.2.2. Blank Removal

To prioritize or identify metabolites from our samples, we need to remove contaminants, i.e., features found in the blanks, before proceeding with statistical analysis[109]. While blank removal can be executed during pre-processing with MZmine 3, which might result in the absence of blank features and samples in both the feature table and metadata, conducting it during post-processing offers more flexibility. If you have performed blank removal during pre-processing, simply skip steps 16-18. Instead, designate the previous variables as the resulting blank-removed table and the metadata for samples, ensuring a seamless continuation of the workflow: `blk_rem <- ft_t` and `md_Samples <- new_md`. For a graphical overview on blank removal, see **Figure 6**, and for more insights, refer to **Box 2**.

Step 15: Examining Metadata Attributes

Run `InsideLevels(new_md)` to identify unique groups within each metadata attribute. This helps to find the attribute column containing sample type information (e.g., 'Blanks', 'Samples').

Step 16: Separating Blank and Sample Files

(User Input Required)

In this step, the data is split into two groups: 'blank' and 'sample' files. It's important to note that 'samples' here include all mzML (or mzXML) files except blanks, including control samples, as they might be influenced by blank features.

- **Identify the Attribute Column:** The user will first be prompted to enter the index number of the attribute containing information about samples and blanks. Here, it is '`ATTRIBUTE_Sample_Type`'.
- **Display Unique Groups:** The unique groups within the chosen attribute column will be displayed. For example, in our dataset, it will show Blank and Sample. However, your dataset might include various groups, such as Blank, Samples, Control, etc.
- **Select the Blank group:** Next, the user will be prompted to enter the index number corresponding to the blank group. If there are multiple groups representing blanks (e.g., Blank, PPL_Blank), their index numbers should be entered, separated by commas.
- **Select the Sample group:** Similarly, the user will be asked to enter the index number(s) for the sample level. If the dataset includes multiple groups for samples (e.g., Sample, Control), the corresponding index numbers should be entered, separated by commas.
- **Subset the Data:** Using the information provided, the metadata (`` `new_md` ``) will be subsetted into `` `md_Blank` `` and `` `md_Samples` ``. The corresponding feature quantification tables will be obtained and named `` `Blank` `` and `` `Samples` ``, respectively.

Step 17: Define Cutoff for Blank Feature Removal

(User Input Required)

In this step, the user will need to set a cutoff value within the range of 0 to 1, with a recommended range of 0.1 to 0.3. This value will determine which features are considered to be artifacts of the blank and thus removed from the dataset. The next step will explain how the features exceeding this cutoff are identified and eliminated.

Step 18: Perform Blank Removal

Calculate the blank's contribution to each feature and eliminate those exceeding the user-defined cutoff. This is achieved by:

- Compute the mean value for each feature within the dataframes (`` `Blank` ``) and (`` `Samples` ``). This step calculates two mean values for each feature, one for blanks and

one for samples. These averages are stored in a new dataframe called `Avg_ft` under the columns 'Avg_blank' and 'Avg_samples'.

- Compute the ratio of the average blank contribution to the average sample contribution for each feature.
- Generate a binary mask where entries corresponding to ratios above the user-defined cutoff are marked as 1 (TRUE), and all others are set to 0 (FALSE). This mask helps in identifying which features are significantly present in blanks as compared to samples.
- Retain only the features associated with 0s in the binary mask. Features with a ratio exceeding the cutoff (marked as 1) are considered artifacts from the blanks and are thus removed. Conversely, if the feature intensity is significantly higher in samples than in blanks, it is deemed a true feature from the samples and is retained (marked as 0).
- The final table, free from blank artifacts, is named `blk_rem`, and its corresponding metadata is `md_Samples`.

The final output is the `blk_rem` table, which excludes background or noise features. Information on the total number of features, the number of background/noise features, and the number of features after noise exclusion is also displayed. Steps 16-18 are displayed in **Box 2**.

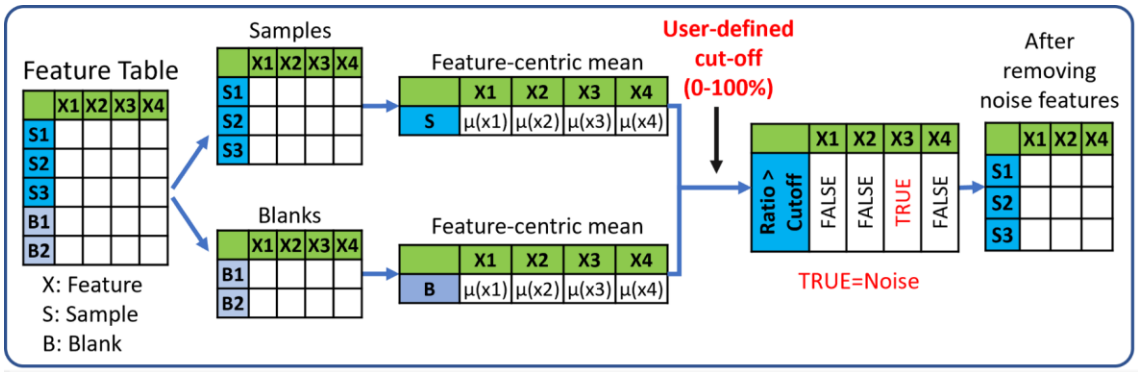
▲**CRITICAL:** Lowering the cutoff to 0.1 demands a greater contribution from the sample (90%) and limits the blank's contribution to 10%. Raising the cutoff leads to fewer background features being identified and more analyte features being observed. Conversely, lowering the cutoff is more stringent and removes more features.

Box 2 - Blank Removal

Some existing methods to achieve blank removal are: creating a molecular network using the online platform, the global natural product social molecular networking (GNPS), and visualizing the network in Cytoscape to manually remove the blank and media nodes. But this is a tedious process[7]; there is also Lawson *et al.*'s msPurity R package with a function called "SubtractMZ" to perform blank removal[110]. Data-adaptive filtering methods have also been suggested to remove features from blanks and low abundant features from samples with undetected values[111].

Another popular feature filtering method is based on the Coefficient of Variance (CV). Also referred to as relative standard deviation (RSD) is a measure of statistical dispersion, calculated as the ratio of the standard deviation to the mean[112]. When pooled QC samples are integrated throughout a study, CV can be used to assess the stability of each feature. As a general rule of thumb, features exhibiting a CV greater than 30% are typically excluded, though the threshold is more stringent (at 20%) for FDA studies. However, it's essential to approach CV filtering with caution. Schiffman *et al.* have highlighted the potential limitations of this method, pointing out that CV primarily evaluates variability across technical replicates without giving weight to biologically meaningful variability across different subjects[113]. Consequently, while CV filtering

might be apt for studies focusing on homogenous samples like plasma or *Escherichia coli* cells, it might not be the best fit for diverse sample sets such as environmental or fecal samples. The dispersion ratio or D-ratio, introduced by Broadhurst *et al*., offers an alternative to a simple CV cut-off by comparing both technical and biological variance. It is calculated by dividing technical variance by the total variance, which includes both technical and biological variances. Therefore, for any feature, a 0% D-ratio signifies that the variance is entirely biological, whereas a 100% D-ratio denotes complete technical noise, without any biological information. So, when assessing D-ratios for metabolites, it is better to retain the ones with D-ratios closer to zero[114].




```
#Getting mean for every feature in blank and Samples in a data frame named 'Avg_ft'

Avg_ft <- data.frame(Avg_blank=colMeans(Blank, na.rm= F))
# set na.rm = F to check if there are NA values. When set as T, NA values are changed to 0

# adding another column 'Avg_samples' for feature means of samples
Avg_ft$`Avg_samples` <- colMeans(Samples, na.rm= F)

#Getting the ratio of blank vs Sample
Avg_ft$`Ratio_blank_Sample` <- (Avg_ft$`Avg_blank`+1)/(Avg_ft$`Avg_samples`+1)

# Creating a bin with 1s when the ratio>Cutoff, else put 0s
Avg_ft$`Bg_bin` <- ifelse(Avg_ft$`Ratio_blank_Sample` > Cutoff, 1, 0 )

#Calculating the number of background features and features present
print(paste("Total no.of features:",nrow(Avg_ft)))
print(paste("No.of Background or noise features:",sum(Avg_ft$`Bg_bin` ==1,na.rm = T)))
print(paste("No.of features after excluding noise:",(ncol(Samples) - sum(Avg_ft$`Bg_bin` ==1,na.rm = T))))

blk_rem <- merge(as.data.frame(t(Samples)), Avg_ft, by=0) %>%
                    filter(Bg_bin == 0) %>% #picking only the features
                    select(-c(Avg_blank,Avg_samples,Ratio_blank_Sample,Bg_bin)) %>% #removing the last 4 columns
                    column_to_rownames(var="Row.names")
blk_rem <- data.frame(t(blk_rem))
```

```
[1] "Total no.of features: 11217"
[1] "No.of Background or noise features: 2125"
[1] "No.of features after excluding noise: 9092"
```

```
head(blk_rem, 2)
dim(blk_rem)
```

	X10015_282.169_2.763_NA	X10035_325.139_2.817_NA	X10037_216.123_2.847_NA	X10047_338.159_2.845_NA	X10058_280.117_2.961_NA
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
SD_01-2018_1_a.mzXML	50907.97	196008.38	90480.91	446560.7	182757.8
SD_01-2018_1_b.mzXML	51443.73	99569.05	411595.38	239022.0	274146.0

180 · 9092

Figure 6: Blank Removal Process: Featuring a graphical representation of the blank removal followed by screenshots of the corresponding R code executed for the procedure.

3.2.3. Imputation

Many feature extraction software programs, such as MZmine 3, often generate tables with missing values denoted as “NA”, “NaN” or 0. This means that for several *m/z* and RT traces in a given sample, there may not be a peak detected and therefore no value is available[76]. However, many statistical approaches, such as Principal Component Analysis (PCA), require numerical values for each observation. Hence, these features with missing values need to be removed or imputed. In this section, we handle the zero values in our blank-removed feature quantification table. Refer to **Figure 7** for a graphical overview on imputation and for more information, see **Box 3**.

Step 19: Analyzing the frequency distribution of relative intensities

We first examine the distribution frequency of the relative intensities in our feature quantification table by creating a histogram. This reveals any notable gaps in the range of values, such as a large number of zeros or a lack of values within a particular range. In our example, we observed many zeros and no values in the range of 0 to 100. The smallest non-zero value in our table was between 100 and 1000.

Step 20: Replacing zeros with random values

We replace all zero values in the dataset with the randomly generated number between 1 and the smallest non-zero value in our blank-removed table. This process, known as imputation, fills in the gaps in our data with plausible values, which can improve subsequent analyses.

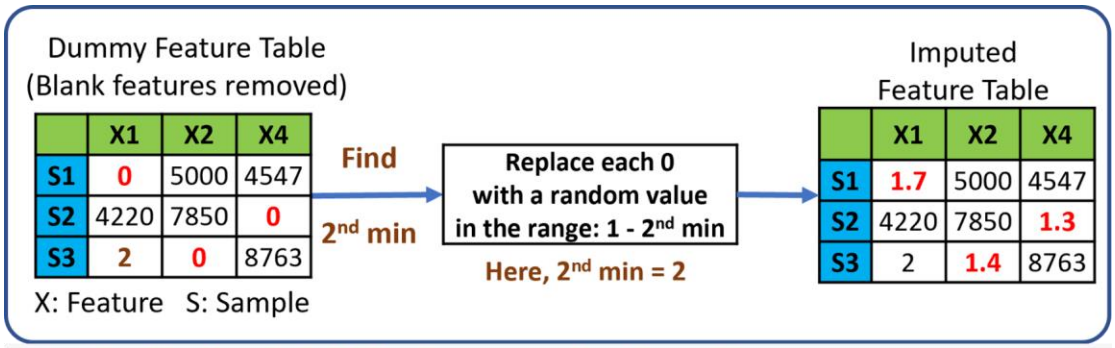
▲**CRITICAL:** Imputation is not advised if one plans to execute a PCoA using the Jaccard distance since Jaccard transforms data into binary (0 and 1). Without zeros, it results in a table full of ones.

Box 3 - Imputation strategies

The appropriate imputation strategy depends on the nature of the missing values:

1. Below the Limit of Detection (LOD): If a value is missing because the corresponding molecule was below the analytical method’s LOD, consider replacing missing values with a low value, ensuring it does not artificially lower the variance[115]. Our imputation method corresponds to this scenario.

2. Sample Processing or Feature Extraction Artifacts: If missing values arise from analysis anomalies (like ion suppression effects or specific retention time shifts) or sample processing artifacts, consider substituting missing values with those similar to values detected in other samples. Here, machine learning methods like k-nearest neighbor (KNN) or random forest (RF) can be useful. KNN fills in multiple missing values by identifying the k nearest data points to a given point[116].




```r
#creating bins from -1 to 10^10 using sequence function seq()
bins <- c(-1,0,(1 * 10^(seq(0,10,1))))

#cut function cuts the give table into its appropriate bins
scores_gapfilled <- cut(as.matrix(blk_rem),bins, labels = c('0','1',paste("1E",1:10,sep="")))

#transform function convert the tables into a column format: easy for visualization
FreqTable <- transform(table(scores_gapfilled)) #contains 2 columns: "scores_x1", "Freq"
FreqTable$Log_Freq <- log(FreqTable$Freq+1) #Log scaling the frequency values
colnames(FreqTable)[1] <- 'Range_Bins' #changing the 1st colname to 'Range Bins'

## GGPLOT2
ggplot(FreqTable, aes(x=Range_Bins, y=Log_Freq)) +
    geom_bar(stat = "identity", position = "dodge", width=0.3) +
    ggtitle(label = "Frequency plot - Gap Filled") +
    xlab("Range") +
    ylab("(Log)Frequency") +
    theme(plot.title = element_text(hjust = 0.5))

Cutoff_LOD <- round(min(blk_rem[blk_rem > 0]))
print(paste0("The limit of detection (LOD) is: ",Cutoff_LOD))
```

```
[1] "The limit of detection (LOD) is: 892"
```

```r
# by setting a seed, we generate the same set of random number all the time
set.seed(141222)

imp <- blk_rem

for (i in 1:ncol(imp)) {
  imp[,i] <- ifelse(imp[,i] == 0,
                    round(runif(nrow(imp), min = 1, max = Cutoff_LOD), 1),
                    imp[,i])
}
```

Figure 7: Imputation Steps: Visual representation of the imputation algorithm complemented by screenshots of associated R code snippets.

3.2.4. Normalization

Sample normalization aims to eliminate systematic bias via adjusting variations across samples[117]. In our pipeline, we show two normalization methods: Total Ion current (TIC) normalization and Probabilistic Quotient Normalization (PQN), implemented using the KODAMA library in our R Notebook. Therefore, we begin this section by installing the KODAMA package.

We recommend that users run both normalization methods and scaling methods (steps 21 to 23), but they can choose either method for further analysis in step 24. Additional information about normalization, including various methods, and guidelines for selecting the most suitable method for a given dataset, is provided in the accompanying **Box 4**. For a graphical view on the provided normalization methods, see **Figure 8**.

Step 21: Total Ion Current (TIC) Normalization

In TIC normalization, also known as total sum normalization, every feature within a sample is normalized relative to the area of the TIC chromatogram[118]. This involves dividing each feature by the sum of peak areas of all features within a sample. The normalization function from the KODAMA package performs row-wise sum operations, and we have the samples arranged in rows and features in columns.

Step 22: Probabilistic Quotient Normalization (PQN)

PQN is another method performed on the imputed table, resulting in a PQN-normalized table with features in columns and samples in rows.

PQN is based on the comparison of a 'test' spectrum (the individual sample to be normalized) with a 'reference' or 'control' spectrum. The steps involved in PQN are as follows[119]:

- **Normalization of Test Spectrum:** The test spectrum is first normalized, typically using a sum normalization technique like TIC.
- **Selection of Control Spectrum:** The control spectrum acts as a standard for comparison. It could be a pre-determined standard obtained from a database or calculated as the mean or median spectrum from all samples or quality control (QC) samples.
- **Calculation of Quotients:** For each sample, quotients are calculated between the features in the test spectrum and the corresponding features in the control spectrum. This step results in a median quotient spectrum for each sample.
- **Normalization by Median Quotient Spectrum:** Each test spectrum is then normalized by dividing it by its corresponding median quotient spectrum. This process scales the test spectrum values relative to the control spectrum, ensuring an equal basis for comparison across all samples.

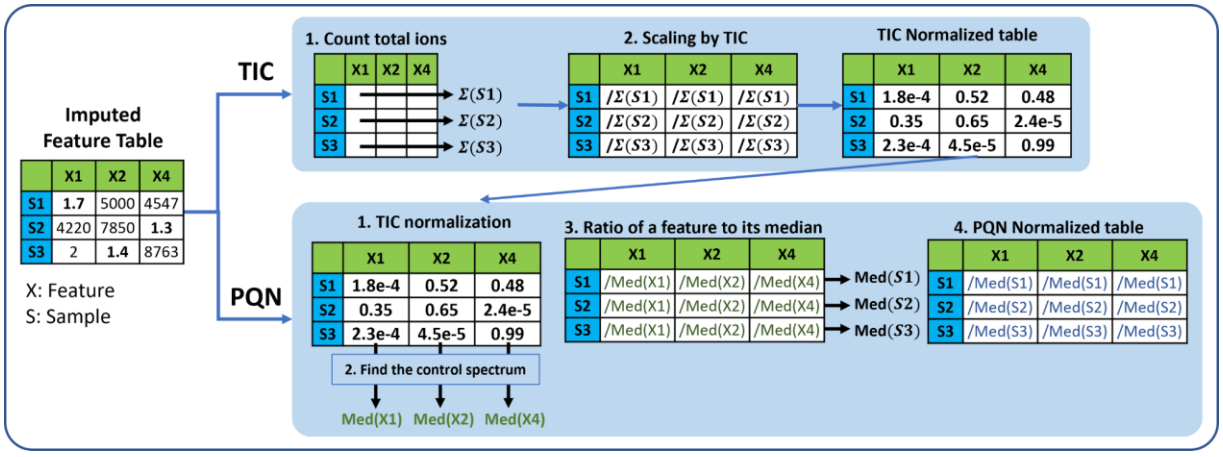
Box 4 - Normalization

Normalization of metabolomics data can rely on either chemical or mathematical strategies. The chemical method, using internal standards and quality controls, is popular in targeted analysis as it effectively balances metabolite concentrations across sample sets and batches. However, for non-targeted metabolomics, mathematical approaches are more popular[117,120]. There are several mathematical normalization methods, each with its strengths and limitations. The selection of a normalization method depends on the specific conditions and requirements of your dataset:

1. **Unit Normalization[121] and TIC Normalization:** Simple and computationally efficient methods useful for large datasets. They equalize

the total sum of signal intensities across each sample. They assume that the abundance of most features does not change significantly across different samples or experimental conditions and their effectiveness decreases with large global changes in metabolite levels (e.g., due to differences in metabolite level such as healthy versus diseased, sample preparation, or instrument sensitivity). TIC normalization might over-correct disease samples with lower intensity reducing the differences between healthy and conditions.

2. **PQN**[119]**:** Recommended when significant size effects are present or when internal normalization disrupts relative peak information[117]. Among several LC/MS-based normalization methods, including Contrast Normalization, Cubic Splines, Cyclic Loess, PQN has been identified as the best performer in reducing sample-to-sample variations[120].
3. **Common Components and Specific Weights Analysis**[122] **(CCSWA)**: A viable alternative when QC and sample data differ.




```
norm_TIC <- normalization(imp, #performing normalization on transformed imputed data
                          method = "sum")$newXtrain
head(norm_TIC,n=3)
dim(norm_TIC)
print(paste('No.of NA values in Normalized data:',sum(is.na(norm_TIC)== T)))
```

```
norm_pqn <- normalization(imp,
                          method = "pqn")$newXtrain
head(norm_pqn,n=3)
dim(norm_pqn)
print(paste('No.of NA values in Normalized data:',sum(is.na(norm_pqn)== T)))
```

Figure 8: Normalization Techniques: Graphical representation of Total-ion-current (TIC) and Probabilistic Quotient normalization (PQN) methods, accompanied by corresponding R code snippets.

3.2.5. Scaling

Scaling methods in metabolomics aim to adjust the range of peak abundances between features[117]. This is done by normalizing the intensities of each feature by a scaling factor, effectively adjusting for fold differences between features[123]. Additional information on scaling

factors can be found in **Box 5** along with the graphical representation of scaling shown in **Figure 9**.

Step 23: Center-Scaling

We apply center-scaling to the imputed data. This allows for a consistent spread of the data, accounting for differences in offset between high and low-abundant features.

In R, the scale function offers different options for centering and scaling data:

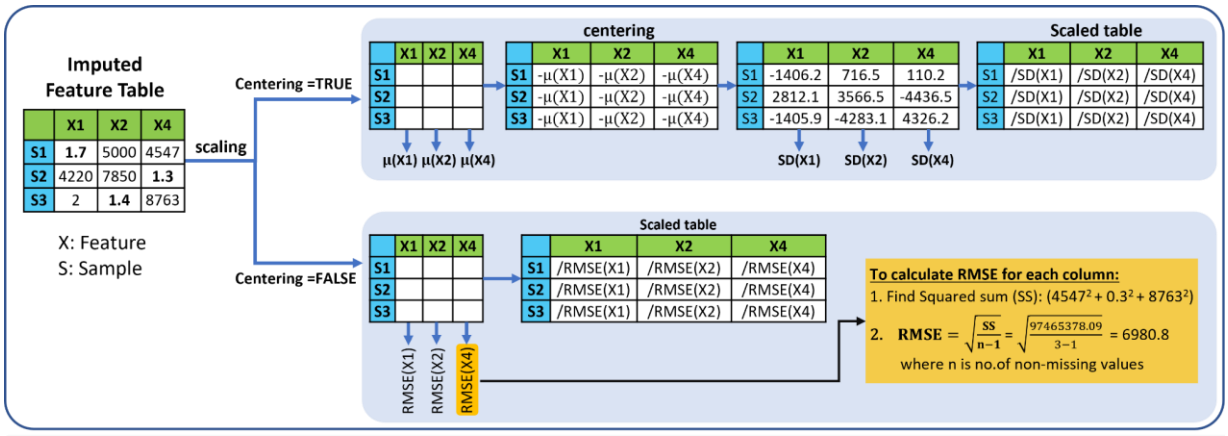
- When center = TRUE, centering is achieved by subtracting the column means (excluding NAs) of the data from their respective columns (each column referring to a feature). Centering ensures that the fluctuations in the data are centered around zero instead of the mean of the metabolite concentrations[123].
- If center = TRUE and scale = TRUE: then scaling is performed by dividing the centered columns by their standard deviations.
- If center = FALSE and scale = TRUE: scaling is done by dividing each column by its root mean square.
- If scale = FALSE, no scaling is performed.

▲**CRITICAL:** Since scaling introduces negative values, trying a PCoA with the Bray-Curtis difference on scaled data will trigger an error.

Box 5 - Scaling

Scaling methods can be categorized into two subclasses based on the scaling factor used[123].

1. **Using data dispersion methods**, such as standard deviation (SD), for scaling: Examples: Autoscaling[124] and Pareto scaling[125]. Autoscaling ensures equal variance (such as SD=1) for each variable, while Pareto scaling uses the square root of SD as the scaling factor.
2. **Using size measures**, such as the mean, for scaling:
   Examples: Level scaling and Poisson scaling. Level scaling converts metabolite concentration changes relative to the mean concentration, while Poisson scaling scales each feature by the square root of the mean[123,126].




```
Imp_s <- scale(imp, center = T, scale = T) # center and scale data
head(Imp_s, n=2)
```

	X10015_282.169_2.763_NA	X10035_325.139_2.817_NA	X10037_216.123_2.847_NA	X10047_338.159_2.845_NA
SD_01-2018_1_a.mzXML	-1.130474	-0.2996814	-0.1117478	2.2186171
SD_01-2018_1_b.mzXML	-1.126516	-0.6037285	1.9885974	0.2986375

Figure 9: Scaling methods: Graphical representation of scaling methods, both centered and non-centered, accompanied by corresponding R code snippet.

Step 24: Choosing data for further analysis

(User Input Required)

Upon executing this step, an overview table is generated, offering a list of the dataframes produced during each phase of data processing along with its respective metadata tables. This includes stages like the initial raw data (Raw Data), post-blank removal data (Blank Removed Data), post-imputation data (Imputed Data), and various normalization stages (TIC Normalized, PQN Normalized, Scaled Data).

To proceed, users must select their dataset of interest by entering the corresponding index number. The chosen dataset will be stored under the `cleaned_data` variable and the corresponding metadata will be taken under the `metadata` variable. These dataframes will be used in subsequent univariate and multivariate analytical steps. This allows the user to:

- **Explore Multiple Datasets:** Easily switch between datasets to examine the effects of different processing steps.
- **Tailor Analyses to Dataset Characteristics:**
  - TIC normalized data is apt for some univariate statistical tests, especially when analyzing the relative abundance of specific features or metabolites across samples without the comparison being skewed by samples that just have overall higher or lower intensities. Also, when using normalized data for multivariate techniques like PCA, it is important to ensure that a few dominant features do not skew the overall results.

- Using scaled data in multivariate techniques like PCA prevents high variance features from dominating. Additionally, machine learning techniques relying on distance measures, like k-means or k-nearest neighbors, benefit from scaled data to ensure uniform feature influence.

However, it is important to note:

- Imputation is not advised if one plans to execute a PCoA using the Jaccard distance since Jaccard transforms data into binary (0 and 1). Without zeros, it results in a table full of ones.
- Since scaling introduces negative values, trying a PCoA with the Bray-Curtis difference on scaled data will trigger an error.

For the purpose of this tutorial, we will use the `scaled_data` as our `cleaned_data` and the respective `metadata` variable is `md_Samples`. However, users are encouraged to experiment with different datasets.

3.3. Multivariate Statistics: ● Timing 50-60 mins

After data cleanup, we will use multivariate statistical analyses to allow for a deeper exploration of samples. The techniques showcased in our workflow are:

- **PCA and PCoA:** Principal Component Analysis (PCA) and Principal Coordinate Analysis (PCoA) are fundamental methods for discerning trends in your data. Coupled with Permutational Multivariate ANOVA **(**PERMANOVA), these techniques enable a comprehensive exploration of sample similarity by calculating correlations or distance matrices.
- **Hierarchical Clustering Analysis (HCA) and Heatmap:** This combination is ideal for hypothesis generation by providing an initial data overview. HCA builds a dendrogram representing the dataset, where individual samples are clustered based on similarity. A heatmap arranged according to the sample or feature similarities defined in the dendrogram creates a clear visual depiction of sample clusters.
- **Supervised Classification Techniques:** We use RF as a key supervised classification technique in this protocol. For advanced users interested in further exploration, additional instructions on XGBoost and hyperparameter tuning are provided in a separate Jupyter Notebook. The link to this additional notebook can be found in the main notebook and the file is available in our GitHub Repository. Additionally, we would like to point to Partial Least Squares - Discriminant Analysis (PLS-DA), another supervised multivariate technique that is frequently used in metabolomics studies simply due to the availability of the model in several software packages and ease of use with default settings. It handles collinear and noisy data well and offers comprehensive results such as classification prediction accuracy, scores and loadings plots. Yet, its prediction accuracy may lag behind methods like RF, especially with datasets handling fewer features. Therefore, PLS-DA might not be suitable for those who want to significantly reduce the feature numbers and then use the model on them[127]. While we do not dismiss the utility of PLS-DA, we suggest

considering alternative models. For a comprehensive comparison of different machine learning-based classification tools, we recommend the study of Mendez *et al*. in which they evaluate eight machine learning algorithms across ten clinical metabolomics datasets for binary classification[128].

Step 25: Installing Packages for Multivariate Analyses ● **Timing 5-10 mins**

To start our multivariate analysis, we first install and load the necessary R packages for this section: “BiocManager”, “ComplexHeatmap”, ggsci”,”dendextend”,”NbClust” and “cowplot”.

3.3.1. PCoA with PERMANOVA

PCoA: Principal coordinates analysis

PCoA is a popular ordination technique used alongside PCA to visualize sample similarities by calculating distance matrices between samples. PCoA groups samples based on their dissimilarity or distances whereas PCA focusses on their correlation or covariance[129]. The process begins by computing a dissimilarity matrix to capture the sample differences. This matrix is then transformed using multidimensional scaling (MDS) to produce a new set of points called Principal Coordinates (PCos) in a lower-dimensional space. The distance between samples in these coordinates reflects the original sample differences[130]. It is important to mention that MDS can be categorized into metric MDS (as in PCoA) and non-metric MDS [129]. In this protocol, we focus solely on metric MDS and more information can be found in **Box 6**. For graphical illustration of PCoA, see **Figure 10**.

Step 26: Prepare Data

Make sure that the metadata (`metadata`) and the feature quantification table (`cleaned_data`) are in the same order. Also, verify that the sample names (row names) in both data tables are identical and in the same order using `identical()` function. It should return TRUE.

Step 27: Calculate Pairwise Distances and Perform PCoA

- Calculate pairwise Euclidean distances across all samples in the feature quantification table using the `vegdist()` function from the ‘vegan’ package[84]. Store the resulting distance or dissimilarity matrix as ‘distm’.
- Apply the `cmdscale()` function from the base R ‘stats’ package to perform MDS on the distance matrix ‘distm’, considering 10 PCos (k=10).

▲**CRITICAL:** The `vegdist()` function offers various methods such as “manhattan”, “euclidean”, “canberra”, “bray”, “jaccard”, “gower”, “binomial”, “chisq” for distance calculation. Using euclidean distance for PCoA is equivalent to performing PCA. However, using `vegdist("euclidean")` and `cmdscale()` cannot provide loadings information. For a comprehensive PCA with both loadings and scores, use the `prcomp()` function such as `pca_result <-`

`prcomp(cleaned_data, center = FALSE, scale. = FALSE)`. Since our `cleaned_data` we use is already centered and scaled, we can set these parameters to FALSE. For loadings and PC scores, you can access `pca_result$rotation` and `pca_result$x` respectively.

Step 28: Analyze PCoA Results

Examine the list generated by the `cmdscale()` function, which includes the following elements:

- 'points' (`PcoA$points`) represents the data matrix with the given PCos
- 'eig' (`PcoA$eig`) indicates the eigenvalues computed for the PCos, which describe the variance explained by each PCo.

Step 29: Plot PCoA Scores

(User Input Required)

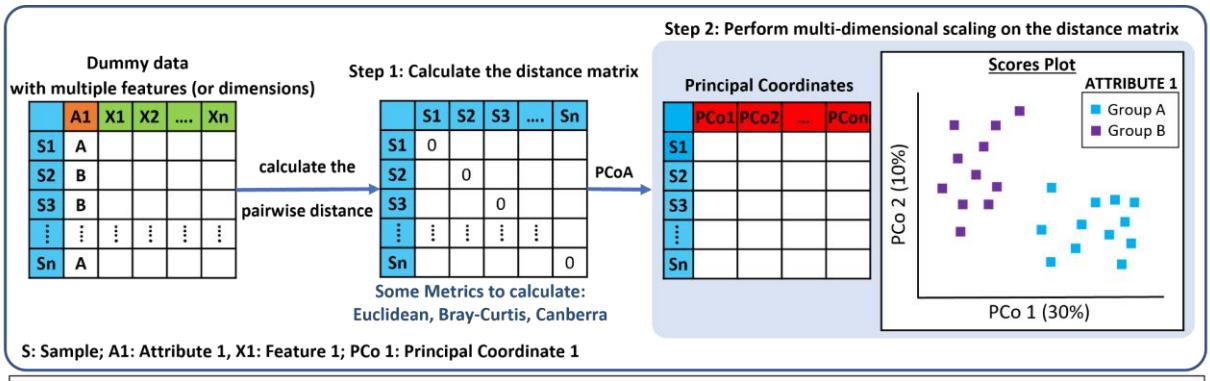
Using the `ggplot2` library, create a PCoA Scores Plot. Here, the samples are color-coded based on the 'ATTRIBUTE_Month' attribute. To view the sample distribution of different attributes, simply adjust the line: `interested_attribute_pcoa = 'ATTRIBUTE_Month'.` Importantly, the aspect ratio of the plot's axes is maintained to ensure accurate representation, in line with recommendations by Nguyen and Holmes[131].

Box 6 - Principal Coordinate Analysis (PCoA)

PCoA offers an advantage over PCA by allowing various distance metrics beyond the Euclidean distance. This flexibility provides different insights into the data pattern based on the chosen dissimilarity measure. For example, when working with categorical data and sparse matrices containing numerous zeros, distance metrics such as Hamming distance and Jaccard distance outperform the Euclidean distance[130,132,133]. Akin to phylogenetic distance measures such as UniFrac distance[134] used in the microbial ecology field, chemical distance matrices are emerging that make use of cosine MS/MS similarity between features[135] or chemical similarity derived from CSI:FingerID[136].

While PCoA effectively reveals chemical trends among samples by working with different distance matrices, it cannot provide direct information about the relationship between features and principal coordinates, unlike PCA which offers 'loadings' information[137]. To discern associated features in such contexts, it is recommended to complement PCoA with other methods like a Heatmap overview, Random Forest analysis or any of the univariate techniques discussed in this protocol.

In addition, to assess the impact of a specific feature on the dispersion of samples along a particular PCoA axis, an indirect analysis can be performed. This involves correlating or regressing the PCoA values of the samples with the corresponding sample scores of the variable of interest[138]. For instance, in our case, to evaluate the influence of Feature 1 on PCo1, we can create a scatter plot by plotting the original values of Feature 1 (sample scores) for all samples against the PCo1 values for all samples. The points on the plot can be colored based on the sampling period. By examining any trends or correlations in the plot, we can observe how the diversity of samples changed during the sampling period.




```
distm <- vegdist(Imp_s, method = "euclidean") #compute distance

#computing multi-dimensional scaling on distance matrix
PcoA <- cmdscale(distm,
                 k = 10, # k=10 gets the first 10 Principal coordinates or PCOs
                 eig = T,
                 add = T)

PcoA_points <- as.data.frame(PcoA$points) #getting the PCOs into dataframe
variance <- round(PcoA$eig*100/sum(PcoA$eig),1) # getting the variance explained by each PCo
names(PcoA_points)[1:10] <- paste0('PCoA', seq(1,10))  #naming the 10 PCos

head(PcoA_points,2) # looking at the 10 PCOs
```

A data.frame: 6 × 10

	PCoA1	PCoA2	PCoA3	PCoA4	PCoA5	PCoA6	PCoA7	PCoA8	PCoA9	PCoA10
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
SD_01-2018_1_a.mzXML	-49.871362	-13.60140	-21.40328	-18.0611465	-3.155365	40.2292107	-22.4594986	14.1888608	29.504792	9.1637140
SD_01-2018_1_b.mzXML	-49.430434	-15.95842	-20.30956	-19.4388300	-3.843483	40.8474155	-22.5707000	11.4764404	31.136861	9.2803551

Figure 10: Principal Coordinate Analysis Overview: The diagram illustrates the process of transforming feature quantification tables into score plots by calculating distance matrices, and plotting principal coordinates. The associated code demonstrates multidimensional scaling using Euclidean distance. Notably, using Euclidean with PCoA is the same as performing PCA; however, the users can adjust to other metrics, like Canberra.

PERMANOVA: Permutational multivariate ANOVA

In multivariate analysis like PCA, it is crucial to measure confidence in observed relationships or separation between objects. This is often achieved via statistical significance tests, which provide a p-value as a measure of the confidence level. For ordination techniques that do not assume a specific data distribution, parametric statistical testing is not applicable[139]. In such cases, resampling methods such as bootstrap, jackknife[140], and permutation tests[141] are used to assess the statistical confidence of the results. These methods generate multiple samples or permutations from original data to estimate variability and assess the significance of observed relationships[139].

Alternatively, non-parametric methods such as PERMANOVA (Permutational Multivariate Analysis of Variance) can be used[142]. PERMANOVA allows for multivariate ANOVA and tests for

differences between object classes. It enables any dissimilarity metric and calculates a test statistic by comparing the dissimilarities between objects within and between classes. Here, the p-values are determined through permutation[139].

Step 30: Testing for Homoscedasticity

(User Input Required)

Before performing PERMANOVA, it is important to validate the homogeneity of group dispersions, often termed as 'Homoscedasticity'. This test ensures that each group exhibits approximately equal variability. Violation of this assumption might inflate the risk of Type I errors (false positives).
If the group dispersions are homogenous, you can proceed with PERMANOVA with greater confidence. However, disparate dispersions require a more cautious interpretation of PERMANOVA results, given their higher susceptibility to Type I errors. In such cases, exploring alternative distance measures, data transformations, or delving into potential biological reasons for the dispersion differences might offer a more comprehensive analysis. To know more about multivariate dispersions, see **Box 7**. For a visual representation of assessing multivariate dispersion and conducting the PERMANOVA analysis in R, refer to **Figure 11**.

Procedure to Evaluate Homoscedasticity:

- As a first step, the user needs to specify the attribute group for assessing group dispersions. Since we are looking for group dispersions, it is important to select a categorical metadata column (for example, '`ATTRIBUTE_Month`') and avoid choosing continuous attributes, such as '`ATTRIBUTE_Injection_order`'.
- Similar to Step 27, we compute a distance matrix ('distm') using the feature quantification table and the selected attribute. For simplicity, we use the Euclidean distance in this instance.
- Using the `betadisper()` function from the vegan package, we evaluate group dispersion against the chosen attribute group.
- The dispersion model is then visualized to offer a clearer perspective.
- Lastly, an ANOVA is executed on the dispersion model. A significant p-value ($P < 0.05$) indicates a violation of the PERMANOVA's foundational assumptions. Conversely, a non-significant result suggests that PERMANOVA is a suitable choice for the given attribute.
- The resulting p-value for '`ATTRIBUTE_Month`' is significant, indicating the presence of group dispersions among different months. This violates the PERMANOVA assumption. When PERMANOVA is performed for this attribute, the PERMANOVA results require a more cautious interpretation.

Step 31: Conduct PERMANOVA Test

- Use the `adonis2()` function from the `vegan` package[84] to conduct a PERMANOVA test. The `adonis2` function allows for the analysis and partitioning of sums of squares using dissimilarity measures.

- Apply the `adonis2` function on the dissimilarity matrix ('distm') and the previously chosen metadata column 'ATTRIBUTE_Month'. This helps in investigating if there are significant differences among the samples collected during three different months.
- Interpret the resulting p-value. In our case, we obtained a p-value of 0.001, indicating a significant difference between the samples.

Box 7 - Dispersion Analysis

In the case of balanced sample sizes across groups, PERMANOVA identifies differences in group centroids, thus reflecting shifts in the multivariate distribution of sample units within the chosen resemblance space. Hence, the type of dissimilarity measure you choose is crucial. For example, unlike Euclidean distance, measures like Jaccard or Bray-Curtis highlight the similarity in species composition and do not focus on the central tendency such as the mean-variance relationship. On the other hand, PERMDISP is specifically tailored to detect variations in multivariate dispersions. Therefore, when analyzing your data, use PERMANOVA to understand group centroid shifts and PERMDISP to evaluate dispersion differences[143].

a)

```
dispersion_model <- betadisper(distm, group)
disp <- anova(dispersion_model)
disp["significant"]  <- ifelse(disp$`Pr(>F)`<0.05, "Significant", "Non-significant")
disp
```

A anova: 2 × 6

	Df	Sum Sq	Mean Sq	F value	Pr(>F)	significant
	<int>	<dbl>	<dbl>	<dbl>	<dbl>	<chr>
Groups	2	37529.38	18764.6917	31.56342	1.890734e-12	Significant
Residuals	177	105227.82	594.5075	NA	NA	NA

b)

```
adonres <- adonis2(distm ~ group)
adonres
```

A anova.cca: 3 × 5

	Df	SumOfSqs	R2	F	Pr(>F)
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
group	2	385128.9	0.236643	27.43527	0.001
Residual	177	1242339.1	0.763357	NA	NA
Total	179	1627468.0	1.000000	NA	NA

Figure 11: Multivariate Dispersion and PERMANOVA Analysis: a) Code snippet for testing multivariate dispersions within the 'group', specifically referencing the 'ATTRIBUTE_Month' column (Dec, Jan, Oct) from the metadata. b) Code snippet for executing PERMANOVA to analyze variations between the aforementioned groups.

Step 32: Define a Function for Streamlined Analysis

To facilitate quicker analysis and avoid rewriting from step 27 to step 31 for testing different parameters, we defined a function, `plotPCoA()`. This function performs a principal coordinates analysis (PCoA) using a chosen distance metric, calculates a PERMANOVA, and plots the results in a 2-D graph. Additionally, it assesses group dispersion prior to the PERMANOVA calculation and displays the significance result in the resulting plot as well.

The function has the following parameters:
- `ft` refers to the desired feature quantification table.
- `md` refers to the respective metadata.
- `distmetric` is the distance metric of choice.
- `category_permanova` is the desired metadata group for PERMANOVA calculation.
- `pcoa_category_type` indicates whether the group type is categorical or continuous.
- `category_pcoa_colors` specifies the metadata attribute for coloring the samples.
- `cols` are the desired colors for the groups.
- `title` is the title of the plot.

Additionally, we have created another simple custom function `save_as_svg()`, to store plots in SVG format utilizing the `svglite` function. This custom function can be used as `save_as_svg(filename, desired_plot, plot_width, plot_height, plot_background)`. Throughout the notebook, you will observe this function being employed post each plot creation to save the visualizations.

Step 33: Applying `plotPCoA()` function on different dataframes

(User Input Required)

In this step, the user can specify the variables as mentioned in the previous step. Here is an example of how to use the `plotPCoA()` function:

```
plotPCoA(
    ft = cleaned_data,
    md = metadata,
    distmetric = "euclidean",
    category_permanova = "ATTRIBUTE_Month",
    pcoa_category_type = 'categorical',
    category_pcoa_colors = "ATTRIBUTE_Month",
    cols = c('orange','darkgreen','red','blue','black'),
    title = 'Principal coordinates plot')
```

Step 34: Get PCoA plots after each data cleanup step

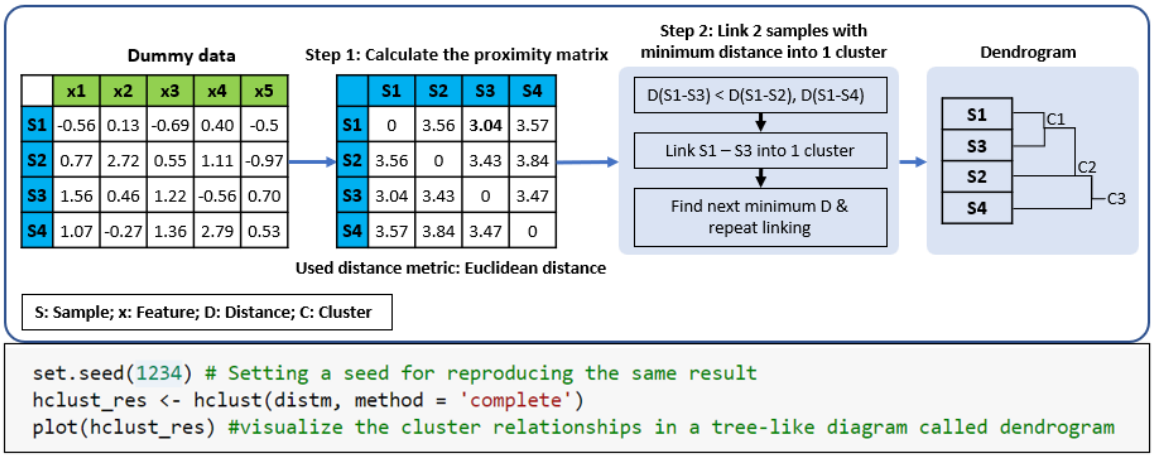
(User Input Required)

In this step, the user can specify parameters such as the distance metric, attribute for PERMANOVA calculation, attribute to color the PCoA scores, the category of the chosen attribute,

similar to the previous plotPCoA step. These inputs will be taken to produce an overview of PCoA plots for all steps of data cleanup.

3.3.2. Hierarchical Cluster Analysis

Clustering, such as Hierarchical Cluster Analysis (HCA), is an unsupervised classification method commonly used in metabolomics to determine the similarity between samples based on their chromatograms or other characteristics. Unlike PCA, which focuses on capturing the maximum variance between samples, clustering aims to group samples with “similar” profiles. The results are often visualized as dendrograms[130] as shown in **Figure 12**.




```
set.seed(1234) # Setting a seed for reproducing the same result
hclust_res <- hclust(distm, method = 'complete')
plot(hclust_res) #visualize the cluster relationships in a tree-like diagram called dendrogram
```

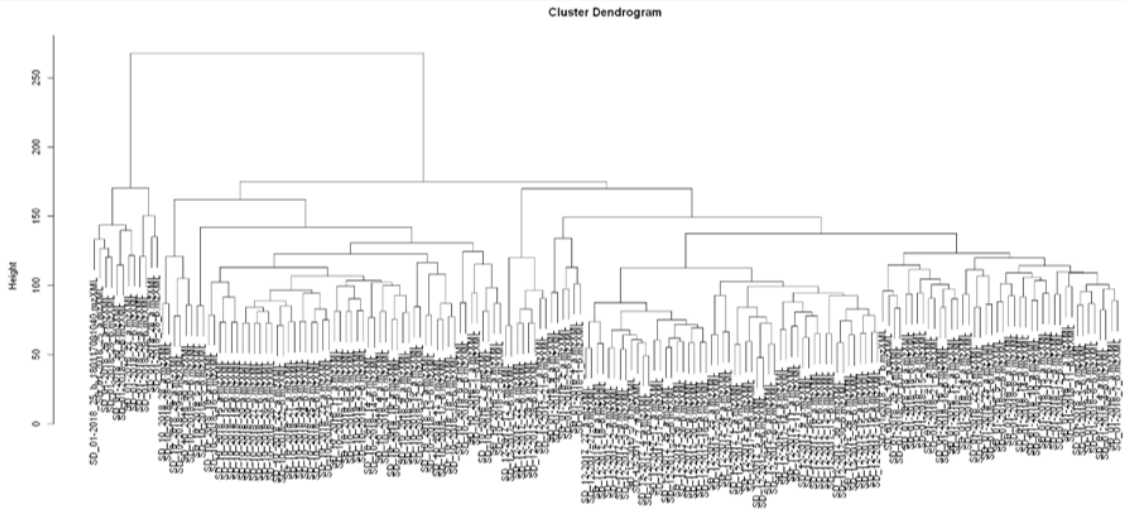


Figure 12: Dendrogram Generation and Analysis: The figure illustrates a dendrogram, as a result of applying HCA to a feature quantification table (e.g., ‘cleaned_data’). From this data, a proximity matrix (or the distance matrix) is calculated (see steps 27 and 30), which subsequently guides the dendrogram creation. Accompanying the illustration is the related code for the cluster generation and dendrogram visualization. The distance matrix ‘distm’ is calculated via Euclidean distance in step 30, though alternative metrics can be chosen by the user. The resultant dendrogram is displayed, initially partitioning samples into two primary clusters: a smaller cluster from a subset of samples (corresponding to samples from January in our example data) and a larger subsequent cluster. Distinct sub-clusters within these main clusters are also discernible.

Step 35: Setting the Plot Size

First, we need to define the size of the output plot, as dendrograms are typically larger in size. Adjust the plot size accordingly to ensure a clear and comprehensible visualization.

Step 36: Executing HCA

Next, we use the `hclust()` function from the 'stats' package to perform HCA. The function is applied to the distance matrix 'distm', calculated based on the feature quantification table ('cleaned_data') using a specified distance metric (e.g., Euclidean, Canberra). The 'method' argument in `hclust()` denotes the linkage method used for measuring the distance between clusters (e.g., complete, single, average). We use the default 'complete' method, which calculates the maximum distance between clusters before combining them. Once HCA is completed, a dendrogram is generated. This dendrogram shows split or merge distances as 'height' along the y-axis, providing a visual representation of the cluster formation.

Step 37: Cutting the Dendrogram

(User Input - Optional)

Similar to k-means clustering, which seeks to establish k clusters with minimum within-cluster variation, we can cut the dendrogram into a specified number of clusters using the `cutree()` function. However, we need to initialize the clustering with random k clusters. For our sample dataset, we define `k=4` with the `cutree()` function, to create four clusters. The user can change the number of clusters.

Step 38: Coloring the Dendrogram

Finally, we can extract the cluster allocation information and color the dendrogram according to the clusters. For our data, the dendrogram suggests two main splits, resulting in four distinct clusters.

Step 39: Determining the Optimal Number of Clusters

Here, we use heuristic methods similar to those applied in k-means clustering to determine the optimal number of clusters. For this purpose, we use the Elbow approach and average silhouette method using the `fviz_nbclust()` function from the 'factoextra' package.

- The Elbow method calculates the total within-cluster sum of squares (WSS) for an increasing number of clusters. WSS signifies the sum of distances between data points and their corresponding centroids within each cluster. Lower WSS values indicate within-cluster variation.
- The resulting Elbow plot presents the WSS on the y-axis and the number of clusters on the x-axis. Lower WSS values suggest minimum within-cluster variation and better clustering. However, the 'elbow' point is considered as an indicator of the optimal number

of clusters, as further cluster additions do not significantly improve the clustering or decrease the WSS. For our example data, this method suggests 3 or 4 clusters. However, defining the 'elbow' can be subjective.

- An alternate approach is the average silhouette method, which assesses clustering quality by determining how well each data point fits within its assigned cluster. In our case, this method proposes two primary clusters.

Both the Elbow and Silhouette methods provide global insights without learning from the data, given their unsupervised nature. But, there are more sophisticated techniques like the gap-statistic which refines the heuristic concepts behind the Elbow and Silhouette techniques and uses a statistical procedure to estimate the optimal cluster count[144]. However, all these methods serve as guidelines rather than definitive answers. In practice, users might choose cluster numbers based on context, for example, in our case with seven sample areas, opting for seven clusters can be insightful. Later, one can check whether these clusters correspond to known sample groups.

3.3.3. Heatmaps

Heatmaps are generally used to visualize complex data or discern patterns across a high-dimensional dataset. They are commonly used in bioinformatics[145], particularly in gene expression analysis and visualizing genomic datasets, owing to their ability to effectively represent thousands of data points[146]. This makes them equally suitable for mass spectrometry-based metabolomic experiments. Heatmaps are efficient in pattern recognition due to their color-coded matrix elements and adjacent dendrograms, which indicate functional relationships between variables and samples[147]. For more information on heatmap, see **Box 8**. To see the resulting heatmap generated by the R code in the Notebook, refer to **Figure 13**. In this section, we will show how to incorporate hierarchical clustering into our heatmap.

Step 40: Preparing Metadata for Heatmap

(User Input Required)

To start with, determine which metadata columns or attributes will be used to decorate the heatmap. In our case, we specified the following attributes: '`ATTRIBUTE_Year`', '`ATTRIBUTE_Month`', and '`ATTRIBUTE_Sample_Area`'. The user can select any number of attribute columns from their metadata as they see fit for the heatmap. A new dataframe is created comprising the chosen metadata.

Step 41: Generating annotations for Heatmap

(User Input - Optional)

For distinct visualization, this step assigns unique colors to each category within chosen attributes from the previous step. We have created a function `generate_colors()`, which utilizes a predefined color-blind-friendly palette of 10 colors to assign colors to these unique groups. Users

can modify these colors if desired. After assigning colors to the subset dataframe, we use this information to decorate the heatmap with annotations from the `HeatmapAnnotation()` function in the 'ComplexHeatmap' package.

Step 42: Creating the Heatmap

(User Input - Optional)

To create the heatmap, apply the Heatmap function from the ComplexHeatmap package on the transposed `cleaned_data` (as previously chosen in step 24). This arranges the features in rows and samples in columns.

- For the heatmap, the color intensity represents the feature intensities, with the intensity scale ranging from 0 (blue) to 1 (dark red), and 0.5 represented as white. This color coding allows for a visual comparison of feature intensity variations across samples.
- The clustering on the y-axis is based on Euclidean distance (`clustering_distance_rows = "euclidean", clustering_distance_columns = "euclidean"`). However, other distance measures such as Manhattan, Minkowski, Canberra, or even Jaccard for binary data, can be chosen based on specific needs.
- The 'complete' linkage method is used for clustering (`clustering_method_rows = "complete",clustering_method_columns = "complete"`).

Step 43: Refining Data Clustering with k-means

Further refine data clustering by incorporating the built-in k-means function within the heatmap as parameters for row and column clustering (`row_km = 5, column_km = 4`). To ensure robustness, perform multiple repeats (`row_km_repeats = 100, column_km_repeats = 100`).

Step 44: Extracting Features from Each Cluster

With a higher number of features, it is difficult to interpret the clustering or labeling of features on the heatmap. To address this, extract the features from each cluster into a separate dataframe. This dataframe containing combined feature names (`XFeatureID_*m/z*_RT_GNPS_annotations) and their respective cluster assignments can be saved as a CSV file for further interpretation. For example, one could merge these cluster assignments with the feature quantification table for import into Cytoscape along with the FBMN and use these cluster assignments for coloring slices in node pie charts.

Box 8 - Heatmap

Although widely used, traditional cluster heatmaps also have limitations. Their data representation in two-dimensional format can be restrictive when processing complex multidimensional data. Furthermore, their static nature does not allow for data to be sorted along different axes, filtered, or focused on specific elements, making the representation of a vast

number of elements quite challenging. Regardless of these limitations, heatmaps are preferred in biological and biomedical data representation because their visual format simplifies data interpretation and comparison. To overcome these limitations, more advanced versions such as XCMS interactive heatmaps are available that offer a more versatile and dynamic data visualization experience[147].

```r
# set the parameters for the type of clustering to perform. You can play with different options
set.seed(1234)
hmap <- Heatmap(
    t(cleaned_data),
    heatmap_legend_param = list(title = "Scaled/centered \n intensity"),
    col = circlize::colorRamp2(c(0, 0.5, 1),
                               colors = c("blue","white",'darkred')),
    show_row_names = FALSE, show_column_names = FALSE,
    cluster_rows = TRUE, cluster_columns = TRUE,
    show_column_dend = TRUE, show_row_dend = TRUE,
    row_dend_reorder = TRUE, column_dend_reorder = TRUE,
    clustering_distance_rows = "euclidean", # you can change the distance here
    clustering_distance_columns = "euclidean", # you can change the distance here
    clustering_method_rows = "complete",
    clustering_method_columns = "complete",
    width = unit(100, "mm"),
    top_annotation = colAnn)

ComplexHeatmap::draw(hmap, heatmap_legend_side="right", annotation_legend_side="right")
```

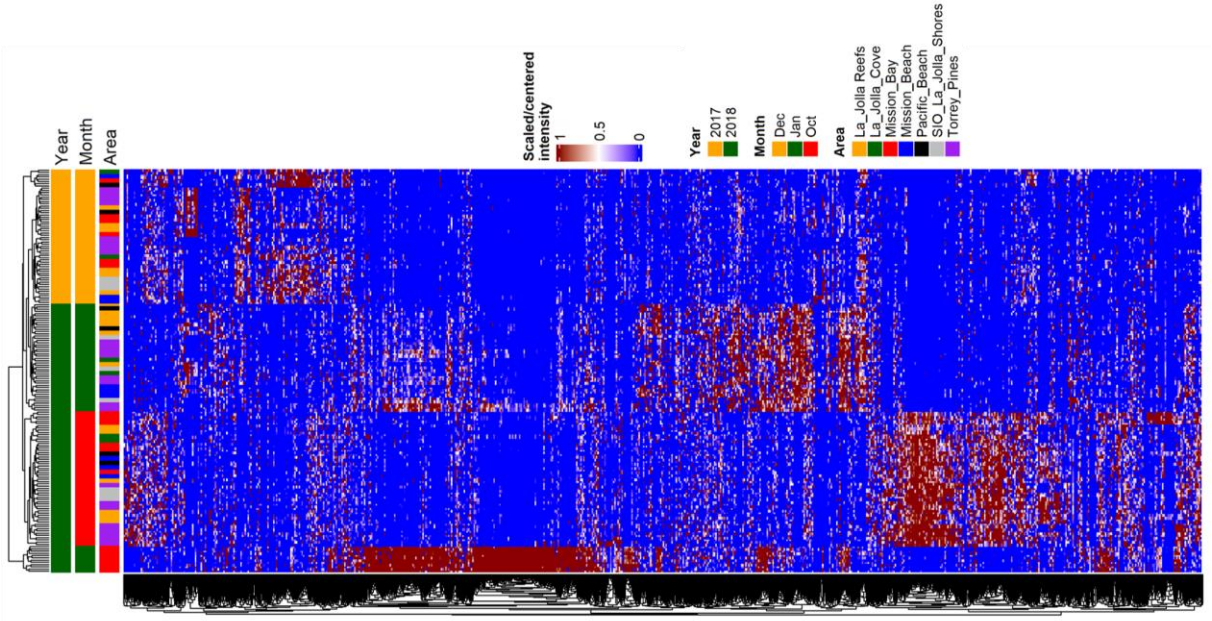


Figure 13: Heatmap Visualization and Construction: This figure presents both the R code snippet used for heatmap creation and the resultant heatmap itself. To facilitate a comprehensive view, the heatmap is oriented horizontally. The feature quantification table used here is the scaled table and feature intensities are color-coded, ranging from blue (0) to red (1). Annotations at the heatmap's top delineate clustering based on variables like year, month, and sample area.

1436

3.3.4. Supervised Classification: **Random Forest**

Unsupervised analysis allows for the discovery of groups or trends in the data without prior assumptions about any predetermined labels or categories, whereas supervised analysis involves the use of labeled data to guide the analysis toward specific objectives such as biomarker discovery, classification, and prediction. In supervised analysis, the algorithm is trained on labeled data to predict the response variable (or dependent variable) based on the predictor variables (or independent variables)[148].

Supervised learning is categorized into classification and regression problems based on the type of response variable: classification for categorical or discrete variables (e.g., cancer vs non-cancer samples), and regression for continuous variables. Popular supervised models in metabolomics include logistic regression, partial least square discriminant analysis (PLS-DA), support vector machines (SVM), k-nearest neighbor (KNN), and random forest (RF). Here, we focused on RF, which offers advantages such as the low risk of overfitting, ease of implementation, interpretability, and minimal hyperparameter tuning requirements[149]. For a detailed overview of random forest, consult **Box 9**.

In our example provided in the notebook, we tried to classify surface seawater samples based on their different sampling sites using random forest. Here, the feature quantification table without metadata is the predictor variable, and the metadata group “Sampling Site” is the response variable. **Figure 14** provides a visualization of the Random Forest algorithm and its implementation in R.

Step 45: Prepare the data for Random Forest

- First, load the `rfPermute` package.
- Start by merging the feature quantification table (in our example, `Imp_s` is chosen as the `cleaned_data` variable) and the corresponding metadata (`md_Samples`) into a dataframe named `cleaned_data_with_md`. This step ensures that the samples are correctly aligned with their corresponding attributes in the metadata, which is essential for the subsequent analyses.

Step 46: Select the Classification Attribute for Random Forest

(User Input Required)

Prepare the dataset used for Random Forest classification so it only contains feature intensity information and attribute of interest for classification. Here, we are classifying the samples according to different sample areas (‘ATTRIBUTE_Sample_Area’). So in this step, the user is prompted to input the index number of the interested attribute to use for the classification.

Step 47: Balance sample sizes

If the sample size varies among the groups, balance the size using the `balancedSampsize()` function.

Step 48: Run Random Forest

Initiate the Random Forest analysis by setting the number of trees and permutations. In our case, we used 500 trees (`ntree`) and 500 permutations (`num.rep`). Here, the primary parameters for Random Forest include the feature quantification table (without the classification data), predictor variable, balanced sample size (`sampsize`), and tree and repetition quantities (`ntree` and `num.rep`).

With the `rfpermute()` function, there is no need for the conventional train-test split, such as the 70-30 or 80-20 ratio. This user-friendly package minimizes the need for parameter tuning. Classification rates in Random Forest rely on out-of-bag (OOB) samples, which are not part of the tree-building process. This eliminates the need to split the dataset into test and train portions, maximizing the amount of information the model has to build a classifier. However, classes with unequal sample sizes, will produce a model that will tend to perform better on the larger class. To alleviate this bias, create a balanced model where the classes are represented by an equal number of samples in each tree and sampling is done without replacement using the `balancedSampsize()` function.

▲CRITICAL:

- Increasing the number of trees and permutations generally enhances the model's performance but also escalates computational costs. It is advised to start with a reasonable number of trees (e.g., 500-1000) and `num.rep` (500-10000), then adjust based on performance.
- When working with large data sets, R may run out of internal memory trying to perform the random forest. To work around this, adding the "`as.factor`" in the predictor variable (y), even if the class is already a factor, will alleviate the memory error.

Step 49: Evaluate model performance

After getting the RF model, we need to evaluate the model's performance using several metrics such as model accuracy, the confusion matrix, trace plot, and check for potential overfitting by comparing testing versus training accuracies.

- The confusion matrix is the most basic summary of a Random Forest. The matrix consists of the 'original class' in rows and the 'predicted class' in columns. The diagonals represent the number of samples correctly classified in each class. The matrix also has columns that show the percent of samples that were correctly classified in a class, along with upper and lower 95% confidence intervals.
- The trace plot shows the OOB (out-of-bag) changes as trees were added to the forest. The model should have enough trees in it so the error rate is stable. If the error rate level increases as the number of trees increases, it may be an indication of overfitting.

Step 50: Interpreting RF Results

Beyond these, the RF results can be interpreted in various ways:

- One could plot the most impactful predictors in the model using violin plots. Here, we show the top 9 predictors
- Compare class predictions versus the actual group in a proximity plot

- Rank features by importance using the 'Mean Decrease Accuracy' metric. This metric helps identify features whose removal significantly impacts the model's accuracy, thus marking their importance. If a feature's removal does not affect accuracy, it may be deemed less important. Features with a 'MeanDecreaseAccuracy.pval' < .05 are considered significant, implying that their absence would affect the model's performance significantly. This ranked list can also be exported as a CSV file for further analysis.

Box 9 - Random Forest

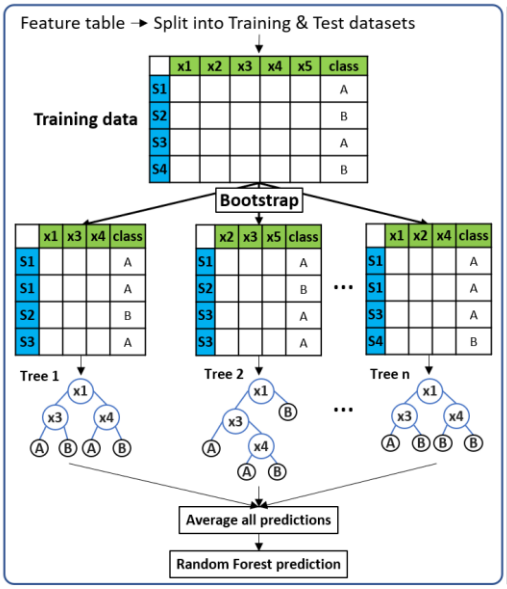
Random Forest (RF) is a powerful machine learning algorithm that operates by dividing data into fractions, building randomized tree predictors on each fraction, and aggregating these predictors together. Generally, RF uses out-of-bag (OOB) error as an estimate of the overall generalization error and obtains variable importance scores through permutation[150].

A unique feature of the RF algorithm is its use of OOB samples, which are the samples not used in the bootstrap sample for a particular tree. Each tree is trained on about two-thirds of the total dataset, with the remaining one-third serving as the OOB samples. The OOB error rate is a measure of prediction accuracy and helps to improve the performance of weak or unstable learners in the model[151].

In RF, variable importance scores are obtained by permuting the values of each variable 'm' within the OOB samples and the tree is used to make predictions on these permuted OOB samples. This essentially disrupts any relationship that variable 'm' might have with the target variable. The model then compares the prediction accuracy on the variable-m-permuted OOB samples to predict accuracy on the original (untouched) OOB samples. The average of the difference in accuracy (between permuted and original OOB) across all trees in the forest gives the raw importance score for variable "m". This raw importance score is often an average value over all trees. To determine if this importance score of variable "m" is statistically significant, a z-score can be calculated by dividing the raw score by its standard error[152].

In RF, there are two common metrics of variable importance used to rank features based on their predictive power: Mean Decrease Accuracy (MDA) and Mean Decrease Gini (MDG). MDA measures the decrease in model accuracy when a particular variable's values are permuted. A large decrease indicates high variable importance; MDG measures how each variable contributes to the homogeneity of the nodes and leaves in the resulting RF. A higher MDG value indicates that splitting the dataset by this variable results in purer nodes. Here, Variable Importance Projection (VIP) could be obtained by normalizing MDA, so they sum to 100, making them more interpretable on a relative scale[153].

Some of the other important parameters to keep in mind to evaluate the performance of the RF model are: model accuracy, confusion matrix (a matrix showing true vs predicted class labels), trace plot, and check for overfitting by comparing testing vs training accuracy. However, supervised models may not be suitable for all data sets, especially those with few observations or unclear class distinctions. Confounding variables, related to both the predictor and response variable, can also make these models unsuitable. For instance, age and gender in a drug study can be confounding variables, leading to erroneous results if not controlled for. In such cases, using supervised models for analysis may not be appropriate.




```
rp <- rfPermute(x = RF_data[,-1], #feature table without the classification info
                y = as.factor(RF_data[,1]), #classification variable
                sampsize=balanced_size, #give the balanced sample size info
                ntree=500,
                num.rep=500,
                proximity = T)
```

```
rp
```

```
An rfPermute model

               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 95
      No. of permutation replicates: 500
                          Start time: 2023-04-03 10:20:49
                            End time: 2023-04-03 11:11:23
                            Run time: 50.6 mins
```

	La_Jolla Reefs	La_Jolla _Cove	Mission_ Bay	Mission_ Beach	Pacific_ Beach	SIO_La_ Jolla_Shores	Torrey_ Pines	pct.correct	LCI_0.95	UCI_0.95
La_Jolla Reefs	27	1	1	1	3	3	0	75.0	57.80	87.9
La_Jolla_ Cove	0	9	0	0	0	3	0	75.0	42.81	94.5
Mission_Bay	5	0	28	2	1	0	0	77.8	60.85	89.9
Mission_ Beach	1	0	0	15	2	0	0	83.3	58.58	96.4
Pacific_ Beach	3	0	1	5	3	0	0	25.0	5.49	57.2
SIO_La_Jolla_ Shores	1	0	0	3	0	10	4	55.6	30.76	78.5
Torrey_Pines	0	0	0	0	0	8	40	83.3	69.78	92.5
Overall	NA	NA	NA	NA	NA	NA	NA	73.3	66.24	79.6

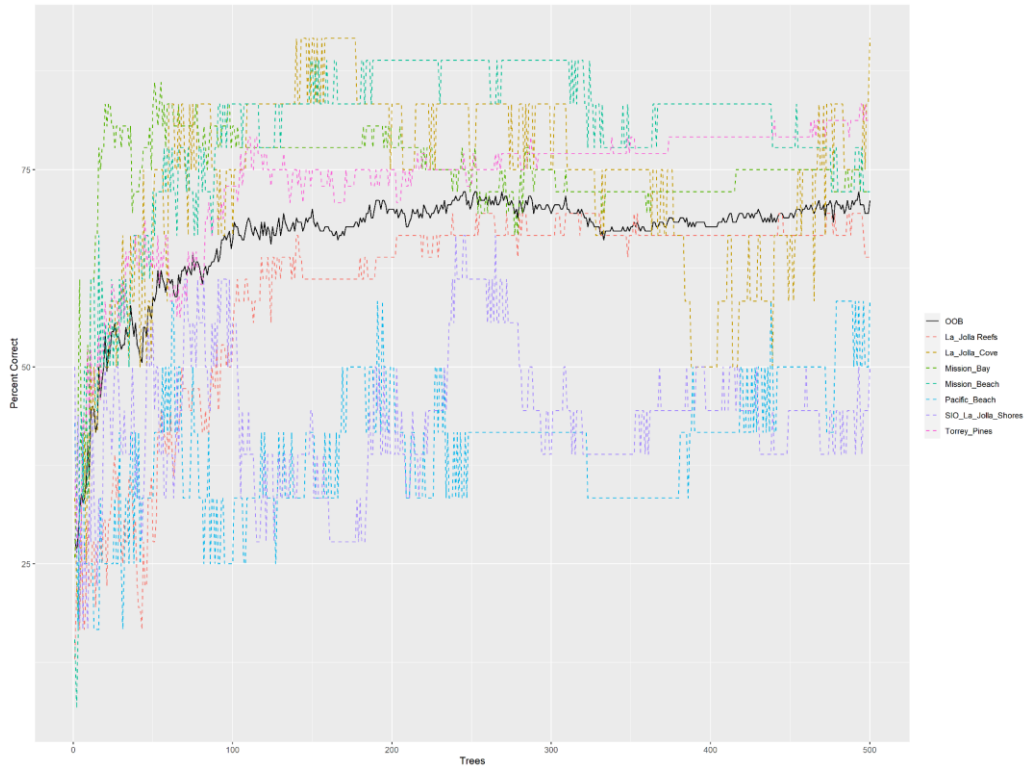


Figure 14: Random Forest Algorithm Visualization and Execution: This figure shows an illustration of the Random Forest algorithm, followed by the code block used for model execution, employing 500 trees and 500 permutations. Outputs of the rfpermute model, including a confusion matrix, are showcased. Additionally, the Out-of-Bag (OOB) error curve, a crucial model evaluation metric, is displayed.

1530

3.4. Univariate Statistics: Timing 50-60 mins

While multivariate analyses offer a comprehensive overview of the data, univariate statistical analyses allow us to focus on specific attributes. Primarily, univariate analysis in metabolomics helps identify individual metabolites that significantly differ between experimental groups, potentially serving as biomarkers for certain conditions or indicators of specific biological processes. It can also reveal impacts on specific metabolic pathways if related metabolites change significantly. However, it is worth noting that univariate analysis does not account for metabolite correlations and interactions, hence, it's best used in conjunction with multivariate analysis for a holistic data interpretation.

For example, our test dataset consists of numerous features collected at seven diverse sample sites. Here, univariate analyses can assess feature differences across these sites. In the case of two site comparisons, the t-test can be used to examine significant feature differences (p value < 0.05). For a comparison involving more than two sample groups, we utilize ANOVA. **Figure 15** provides a flowchart that guides the selection of appropriate statistical tests based on data normality and homogeneity. In the event of significant differences, we represent these findings through a bar graph that captures the distribution of a 'significant' feature across sample conditions. Post-hoc tests are also introduced as supplementary tools to identify which groups' average values significantly differ.

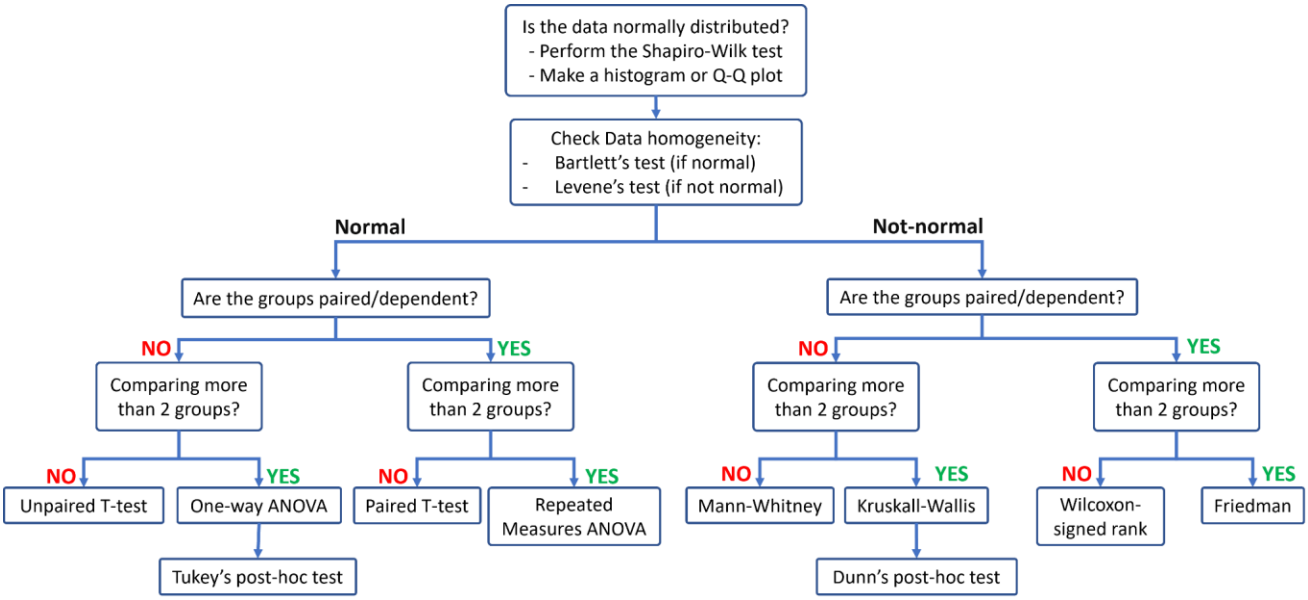


Figure 15: Flowchart detailing the selection of statistical tests for univariate analysis, based on data normality and homogeneity.

When conducting multiple univariate tests simultaneously, as is common in metabolomics, there is an increased risk of false positives. To manage this, the False Discovery Rate (FDR) gauges the expected false positives among significant results. While the classical Bonferroni correction addresses false positives, it could increase the false negative rate. The following are some advanced methods that focus on maximizing true discoveries without escalating the false positive error rate[154].

- **Benjamini-Hochberg (BH):** Commonly used in metabolomics for being less conservative than Bonferroni. It ranks p-values and adjusts them, targeting the expected false positives among all positives, rather than across all tests. It calculates FDR as Expected (False Positive/ (False Positive+True Positive)).
- **Benjamini-Yekutieli (BY):** An iteration of BH that is suitable when tests have dependencies.
- **Storey's q-value:** This approach estimates the proportion of true null hypotheses (i.e., no effect) among all hypotheses and then computes a q-value for each test, which is the FDR analogue to the p-value[155].

In metabolomics, it is crucial to apply FDR correction methods to univariate results to ensure that the identified significant metabolites are not just statistical artifacts but reflect genuine biological differences. In all our univariate tests, we apply the BH metric to our p-values.

Step 51: Install Packages for Univariate Analyses ● Timing **5 mins**

Start by installing the packages necessary for this section: FSA[89] (v0.9.4), matrixStats[90] (v0.63.0).

3.4.1. Test for Normality

Testing for normality is often one of the first steps in univariate analysis and is crucial in deciding whether to use parametric or non-parametric tests. Parametric tests like t-test or ANOVA assume data follows a normal distribution, characterized by a symmetric bell-shaped curve with two key parameters: mean and standard deviation. Thus, before applying any statistical test, it is common to evaluate for normality with tests such as the Shapiro-Wilk test or the Kolmogorov-Smirnov test. Notably, Shapiro-Wilk is more suitable for small sample sizes (N < 50). Here, "normal" applies to the entire population, and not just the sample data. The resulting 'p value' from these tests only indicates the probability of the data to be sampled originating from a normal distribution. A graphical representation of testing normality of features is shown in **Figure 16**. Normality becomes less critical with large samples due to the Central Limit Theorem. In such cases, parametric tests can still be applied regardless of the normality. When the data does not follow a normal distribution, one can follow non-parametric tests, such as the Mann-Whitney U test or the Kruskal-Wallis test[156]. In addition to this, to know more on normality assumptions, refer to **Box 10**.

In our pipeline, we conduct a normality test using two approaches: visual representations such as histograms and quantile-quantile plots (Q-Q plots), and the Shapiro-Wilk statistical test.

Step 52: Normality Testing for One Feature

To illustrate how to test for normality, pick one feature and generate a Q-Q plot using the `qqnorm()` and `qqline()` functions. Then, perform a Shapiro-Wilk test using the `shapiro.test()` function. Additionally, demonstrate how log-transforming the data can improve normality.

Step 53: Normality Testing for All Features

Perform a Shapiro-Wilk test for each feature and record the resulting p-values. Correct these p-values for false discovery rate (FDR) using the Benjamini & Hochberg method. If the adjusted p-value ('`p_adj`') is less than 0.05, reject the null hypothesis and consider the data to be non-normal. Tally up the features that fall under normal and non-normal distributions. If the majority of features are non-normal, consider using non-parametric tests for further analysis.

Box 10 - Normality assumptions

Besides normality, it is essential to consider two other critical assumptions when deciding between parametric and non-parametric tests: homogeneity of variances (homoscedasticity) and independence. Homoscedasticity demands that within-group variances are equal. If unequal (heteroscedasticity), it increases the chance of falsely identifying a "significant" result. Homoscedasticity can be evaluated graphically via boxplots or statistically via Levene's and Bartlett's tests. Here, the null hypothesis (H0) for these tests states that the within-group variances are equal. If the p-value is less than 0.05, it indicates a difference in population variances. The final assumption, 'independence', stipulates that the occurrence of one event does not influence the probability of another. In a metabolomic context, this implies that knowledge of one sample value does not predict another's. However, these assumptions, particularly normality, are seldom fully met in real-world metabolomics datasets[157,158].

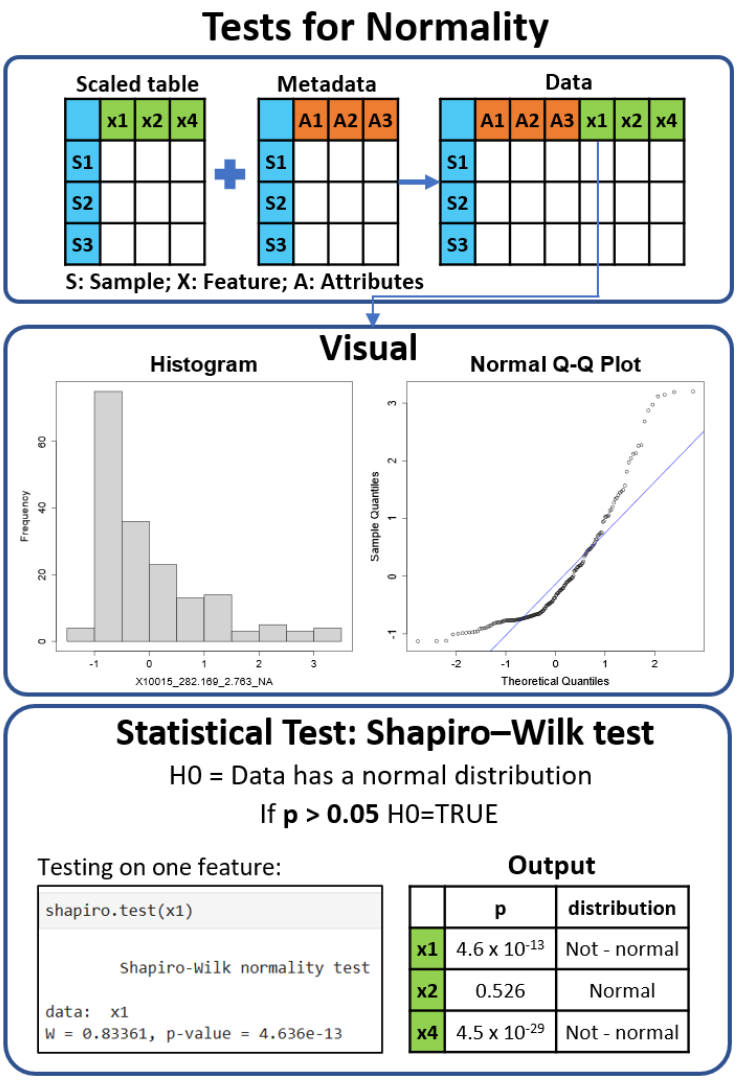


Figure 16: Assessing Normality of Features: This figure illustrates methods to assess normality for individual features. It showcases visual approaches like histograms and Q-Q plots, where deviations from normality can be visually assessed. The third segment delves into significance testing using the Shapiro-Wilk test, emphasizing that a p-value greater than 0.05 suggests a normal distribution.

3.4.2. Parametric tests

3.4.2.1. ANOVA test

The analysis of variance (ANOVA) is the statistical procedure used to test if there exists a significant difference in the means of a dependent variable between three or more groups. As opposed to a pair-wise comparison where we compare the means in a variable (i.e., $\mu_1=\mu_2$), in the ANOVA we compare the means of several groups[159]. For a deeper understanding of ANOVA, please refer to **Box 11**. Furthermore, **Figure 17** offers a visual explanation of the ANOVA

algorithm, detailing both the R code and a resulting plot that contrasts the F-statistic with p-values, highlighting significant features.

Step 54: Running ANOVA on one feature

(User Input Required)

Here the user is prompted to enter the index number of the attribute for performing ANOVA. In the tutorial, we use '`ATTRIBUTE_Sample_Area`'. The resulting ANOVA statistics are shown in a table format.

Step 55: Running ANOVA on all features

- For each metabolite feature, execute an ANOVA test within a for loop. The output for each feature is stored in a dataframe named `` `anova_out` ``. The 'for loop' passes each feature column as the first argument of the `aov()` function against the selected attribute from the previous step ('`ATTRIBUTE_Sample_Area`'). This is because we are examining how a particular feature varies across different sample areas.
- Tidy up the ANOVA output for each feature into a table using the `tidy()` function from the broom package.
- Out of the two rows in the ANOVA summary table, select only the first row of this table (which contains the means, F-statistic, and p-value) and leave the second row consisting of the residuals.
- Consolidate these rows into a single dataframe which contains the features, their corresponding p-values, their BH-corrected p-values, and their significance status in several columns. Features with a BH-corrected p-value ('`anova_out$p_BH`') less than 0.05 are considered significant.

Step 56: Subsetting Significant Features

Filter out the significant features for further examination. Display the count of significant and non-significant features.

Step 57: Visualize ANOVA Results

Sort the `` `anova_out` `` results by p-value and visualize the significant features using `ggplot()`. This involves plotting log-transformed F-Statistic values on the x-axis against negative logarithm of `` `p_BH` `` values on the y-axis. As F-Statistic and p-values can vary greatly, their log values offer easier visualization. To prevent clutter, limit the display to the names of the top 6 significant features.

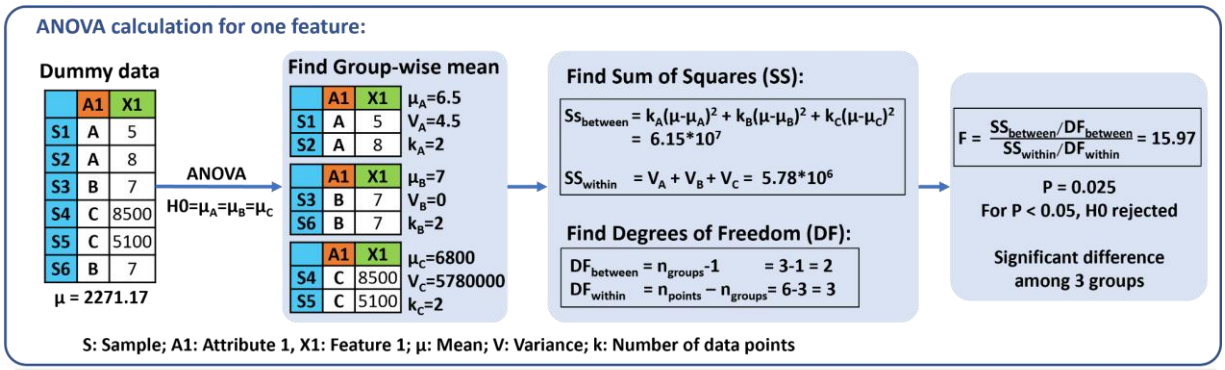
Step 58: Visualize Top Significant Metabolites

Generate boxplots for the top 4 significant metabolites to observe how their intensity levels differ across sampling sites. Extract these metabolites' data from the `` `uni_data` `` dataframe, which

contains both feature intensities and metadata, and plot their intensities based on the sampling sites. In our example, the higher intensities of these features in the 'Mission Bay' sample area primarily account for the observed differences between sampling sites.

Box 11 - ANOVA

If a pairwise test is used (e.g., a t-test), an increased probability of getting a false positive difference (Type I error) would be observed just by chance due to the effects of multiple comparisons[160]. Instead, in the ANOVA test we can perform a single test to see if the observed differences are due to randomness or due to the grouping of the samples (e.g., origin, location, type of soil, etc.). The F-statistic is calculated using the sum of squares and the degrees of freedom (see **Figure 17**) and compared to a standard F-distribution to determine whether the differences among group means are greater than would be expected by chance. Importantly, the alternative hypothesis (i.e., where a difference exists between the means) is unspecific. This means that the test does not tell us where the difference(s) lie (e.g., if the difference is $\mu_A \neq \mu_B$ or $\mu_B \neq \mu_C$), it only tells us whether there exists a difference among all the means. The first assumption of the ANOVA test is the normality of population distribution and the homogeneity in their variances[157,161]. Non-parametric tests should be used if these assumptions do not hold in the data of interest.




```
anova_group <- uni_metadata[,interested_attribute_anova]
anova_group <- as.factor(anova_group) # convert the attribute to 'factor' type

broom::tidy(aov(uni_data[,1] ~ anova_group)) #tidy summarizes the anova ouptut in a  tibble
```

A tibble: 2 × 6

term	df	sumsq	meansq	statistic	p.value
<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
anova_group	6	7.560749	1.2601248	1.271597	0.2728111
Residuals	173	171.439251	0.9909783	NA	NA

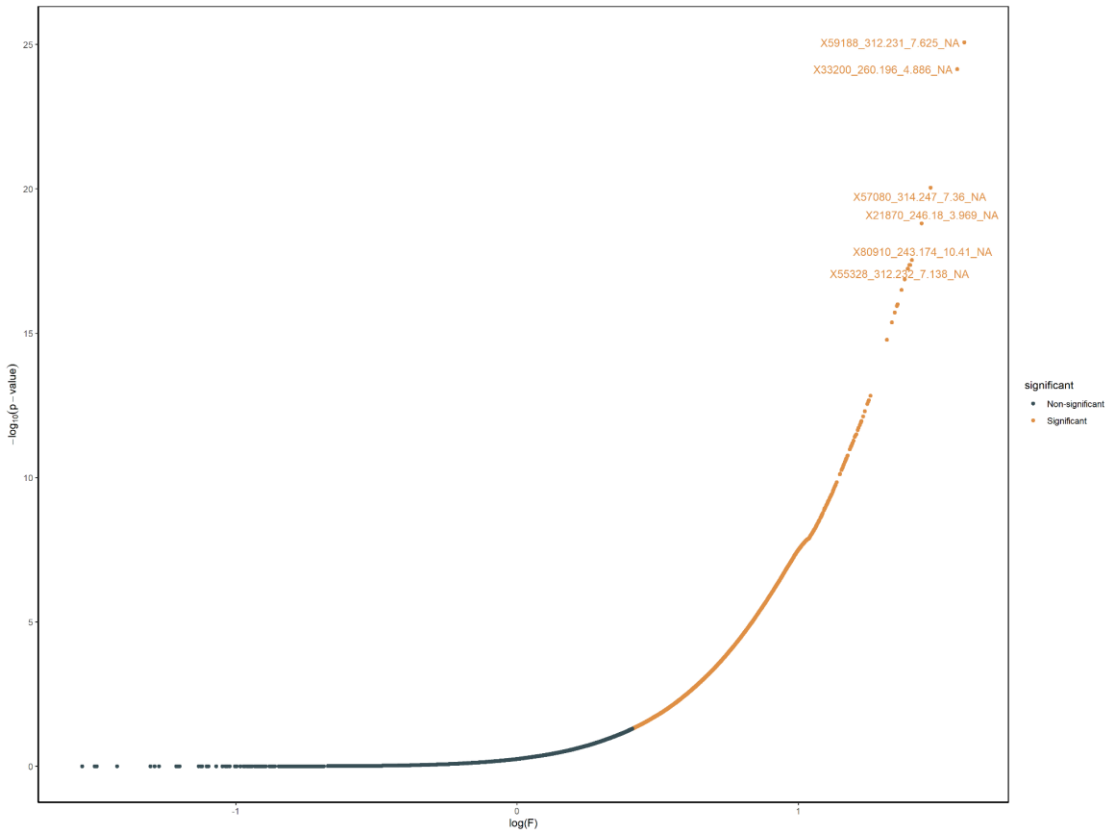


Figure 17: ANOVA Overview: The illustrative figure depicts the ANOVA process applied to a sample feature across groups A, B, and C using dummy data. Following this, the code block is used to test ANOVA on our dataset. This process involves selecting metadata for grouped information (e.g., sample areas), factorizing it for grouping, and then presenting the ANOVA outcome for one of the features in relation to various sample areas. A complementary volcano plot showcases the significance of features by mapping log(F-statistic of ANOVA) against negative logarithm of p-value.

3.4.2.2. Tukey's Honestly Significant Difference (HSD) test

If the ANOVA test provides evidence that a difference indeed exists between the means of the groups, the next step is to find between which groups the difference or differences exist. To do this, we can conduct a Tukey HSD post hoc test used to compare multiple means in a single analysis[157]. Refer to **Box 12** for more information on Tukey's test. Additionally, **Figure 18** provides a visual guide for applying the Tukey test, its implementation in R, and a resulting volcano plot that highlights significant features from our pairwise comparison.

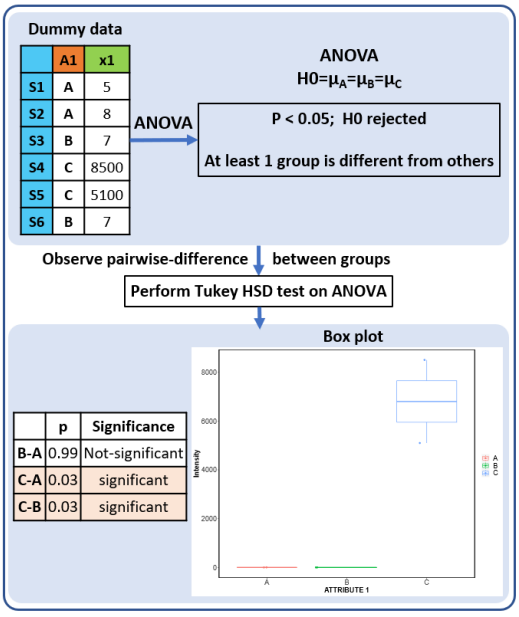
Step 59: Perform Tukey HSD for a Significant Feature

First, we select a feature identified as significant in the ANOVA result, using `anova_sig_names` generated in step 56. From the ANOVA output, we subset the data for

this significant feature and conduct a Tukey HSD test. The output is a comprehensive table providing an assessment of every possible pairwise group difference as shown in the figure.
To conduct a Tukey HSD test for all features, consider specifying just a one-pair comparison to maintain simplicity. For instance, based on the ANOVA results, the sampling site 'Mission Bay' appeared to significantly differ from others for the top four metabolites, hence we can focus on the results from comparisons between 'Mission Bay' and another specific sampling site in the subsequent step.

Box 12 - Tukey's post hoc test

One of the goals of this test is to overcome the Type I error rate inflation of doing multiple comparisons[157]. The most used post hoc test for ANOVA is Tukey's Honestly Significant Difference (HSD). To calculate the HSD between two means, a statistical distribution defined by Student (called the q distribution) is used which takes into account the number of means being compared[162].



	A1	x1
S1	A	5
S2	A	8
S3	B	7
S4	C	8500
S5	C	5100
S6	B	7

	p	Significance
B-A	0.99	Not-significant
C-A	0.03	significant
C-B	0.03	significant

```
model_1 <- anova_model[[anova_sig_names[1]]] #looking at one of the anova model that showed significant difference
broom::tidy(TukeyHSD(model_1)) # Perform Tukey HSD test on the model_1 and summarize the result
```

A tibble: 21 × 7

term	contrast	null.value	estimate	conf.low	conf.high	adj.p.value
<chr>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
Data$ATTRIBUTE_Sample_Area	La_Jolla_Cove-La_Jolla Reefs	0	-0.0146107707	-0.6740913	0.6448698	1.000000e+00
Data$ATTRIBUTE_Sample_Area	Mission_Bay-La_Jolla Reefs	0	1.9009769489	1.4346538	2.3673001	0.000000e+00
Data$ATTRIBUTE_Sample_Area	Mission_Beach-La_Jolla Reefs	0	-0.0134035939	-0.5845305	0.5577233	1.000000e+00
Data$ATTRIBUTE_Sample_Area	Pacific_Beach-La_Jolla Reefs	0	0.0642958118	-0.5951848	0.7237764	9.999494e-01
Data$ATTRIBUTE_Sample_Area	SIO_La_Jolla_Shores-La_Jolla Reefs	0	-0.0132175276	-0.5843445	0.5579094	1.000000e+00
Data$ATTRIBUTE_Sample_Area	Torrey_Pines-La_Jolla Reefs	0	0.0317892050	-0.4044162	0.4679946	9.999910e-01
Data$ATTRIBUTE_Sample_Area	Mission_Bay-La_Jolla_Cove	0	1.9155877196	1.2561071	2.5750683	1.142419e-13
Data$ATTRIBUTE_Sample_Area	Mission_Beach-La_Jolla_Cove	0	0.0012071769	-0.7361145	0.7385289	1.000000e+00
Data$ATTRIBUTE_Sample_Area	Pacific_Beach-La_Jolla_Cove	0	0.0789065825	-0.7287889	0.8866020	9.999488e-01
Data$ATTRIBUTE_Sample_Area	SIO_La_Jolla_Shores-La_Jolla_Cove	0	0.0013932431	-0.7359285	0.7387149	1.000000e+00
Data$ATTRIBUTE_Sample_Area	Torrey_Pines-La_Jolla_Cove	0	0.0463999757	-0.5921393	0.6849393	9.999911e-01
Data$ATTRIBUTE_Sample_Area	Mission_Beach-Mission_Bay	0	-1.9143805428	-2.4855075	-1.3432536	4.685141e-14
Data$ATTRIBUTE_Sample_Area	Pacific_Beach-Mission_Bay	0	-1.8366811371	-2.4961617	-1.1772006	6.189493e-13
Data$ATTRIBUTE_Sample_Area	SIO_La_Jolla_Shores-Mission_Bay	0	-1.9141944766	-2.4853214	-1.3430675	4.718448e-14
Data$ATTRIBUTE_Sample_Area	Torrey_Pines-Mission_Bay	0	-1.8691877439	-2.3053931	-1.4329823	0.000000e+00
Data$ATTRIBUTE_Sample_Area	Pacific_Beach-Mission_Beach	0	0.0776994056	-0.6596223	0.8150211	9.999201e-01
Data$ATTRIBUTE_Sample_Area	SIO_La_Jolla_Shores-Mission_Beach	0	0.0001860662	-0.6592945	0.6596666	1.000000e+00
Data$ATTRIBUTE_Sample_Area	Torrey_Pines-Mission_Beach	0	0.0451927988	-0.5016196	0.5920052	9.999809e-01
Data$ATTRIBUTE_Sample_Area	SIO_La_Jolla_Shores-Pacific_Beach	0	-0.0775133394	-0.8148350	0.6598084	9.999212e-01
Data$ATTRIBUTE_Sample_Area	Torrey_Pines-Pacific_Beach	0	-0.0325066068	-0.6710459	0.6060327	9.999989e-01
Data$ATTRIBUTE_Sample_Area	Torrey_Pines-SIO_La_Jolla_Shores	0	0.0450067326	-0.5018057	0.5918191	9.999814e-01

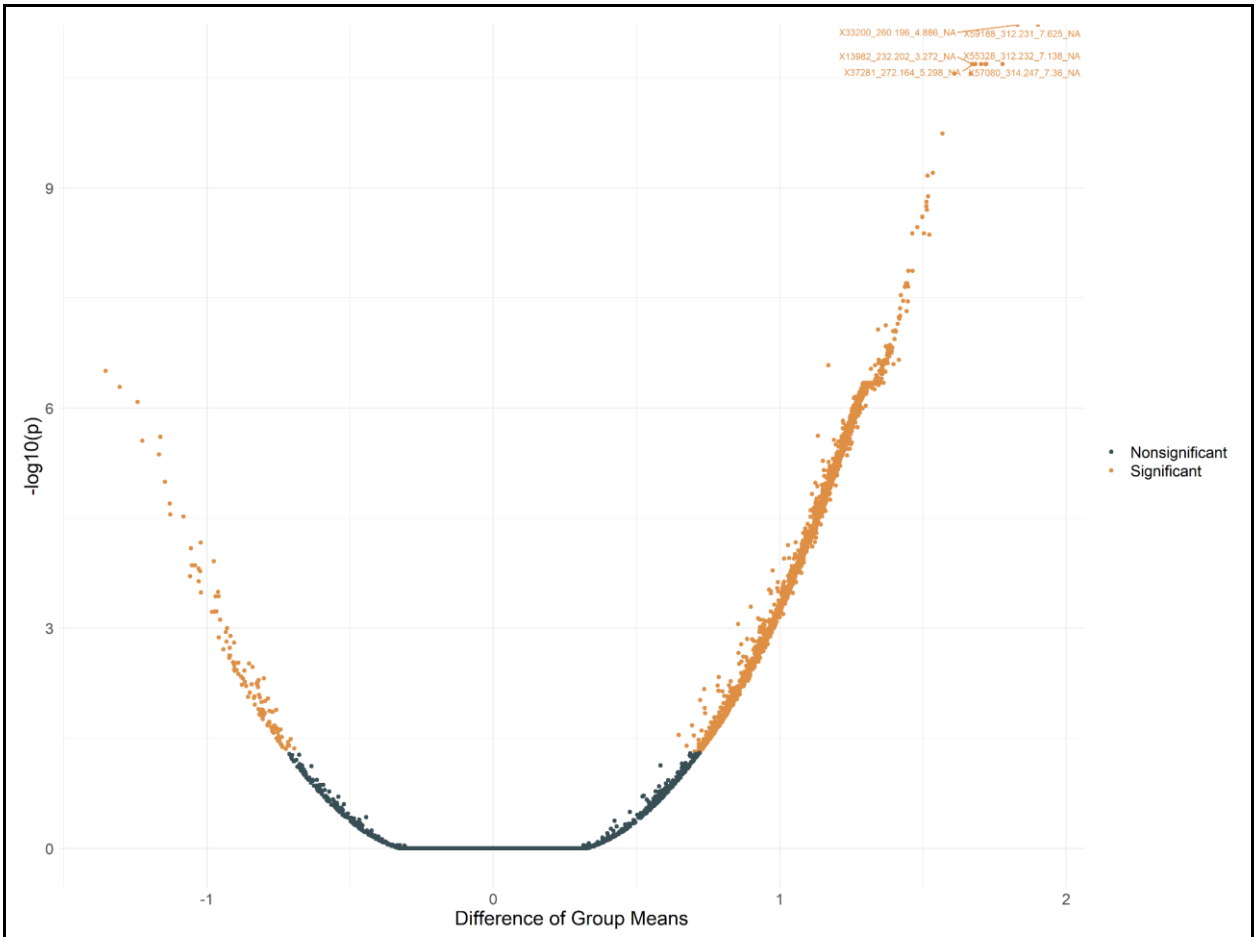


Figure 18: Overview of Tukey's HSD Test: This figure starts with an illustration that showcases how ANOVA's significance suggests that at least one group differs significantly from others, which then necessitates a further analysis through pairwise comparisons using Tukey's HSD test. Alongside, we present the code block demonstrating the Tukey test applied to the first significant feature identified via ANOVA. Given the presence of 7 sample areas, the output presents p-values for all potential 21 pairwise comparisons. Having executed this for all ANOVA-significant features, we particularly highlighted comparisons between 'Mission Bay' and 'La Jolla Reef'. The resulting significance is visualized via a volcano plot, where right-tailed features exhibit higher prevalence in 'Mission Bay', while left-tailed features dominate in 'La Jolla Reef'.

Step 60: Perform Tukey HSD for All Significant Features

(User Input Required)

Carry out a Tukey HSD test for all the significant features identified in the ANOVA. Then, filter the results for the specific comparison such as 'Mission Bay vs. La Jolla Reefs'. Here, users are prompted to input the index number corresponding to their desired comparison from the 'contrast' column displayed in the previous step's output. As a result of the Tukey test of this pairwise interaction, p-values are produced for each feature. After applying the BH correction method, features with corrected p-values `(output_tukey$p_BH < 0.05)` are highlighted as significantly different between the selected sites.

Step 61: Count Significant Features

Determine how many features exhibit a significant difference between the chosen sites and how many do not.

Step 62: Visualize Results with a Volcano Plot

Create a volcano plot with '`-log(p_BH)`' on the y-axis and the group difference ('`estimate`') on the x-axis. Display the names of the top findings on the plot to highlight the most significant differences between the chosen sites. Additionally, visualize the top 2 significant metabolites as boxplots from both extremes of the volcano plot (right and left tips) to clearly represent if the significant metabolite is upregulated or downregulated among the chosen sites.

3.4.2.3. T-tests

Step 63: Select Attribute for T-Test Analysis

(User Input Required)

A t-test is suitable for comparisons involving just two groups. Therefore, users should specify the attribute for the two distinct groups by providing the corresponding index number. For our example, we explore the metabolome's response to rainfall. Hence, we introduce an '`ATTRIBUTE_rainfall`' column, designating '1' for 'Jan-2018' (a high rainfall period) and '0' for the remaining months.

▲CRITICAL: This column addition caters to our dataset's context. Users with pre-existing binary attributes can skip this addition, while others may adjust this step to align with their data.

Step 64: Perform T-Test

Following the same steps as ANOVA (from steps 54 to 57), the `t.test()` function is used in place of `aov()` in this case. The final output is a dataframe 'ttest_output' containing the significance of each feature for the two conditions under investigation.

Step 65: Plot T-Test Results

Visualize the t-test results using a volcano plot, with the '`estimate`' (difference in means of the two conditions for each feature) on the x-axis and '`-log(p_BH)`' on the y-axis. Additionally, visualize the top 2 significant metabolites as boxplots from both extremes of the volcano plot (right and left tips) to clearly represent if the significant metabolite is upregulated or downregulated for the chosen attribute.

▲CRITICAL: Unlike ANOVA, post-hoc tests are not needed for t-tests as there are only two conditions to compare. In ANOVA, when a feature is found to be significant, post-hoc tests help determine which specific groups show significant differences.

For all the above tests, the respective significance values can be saved as a CSV table, and the plots can be saved in SVG, PDF, or PNG formats for further analysis or presentation.

3.4.3. Non-Parametric Tests

3.4.3.1. Kruskal-Wallis Test

The Kruskal-Wallis test is a non-parametric statistical test used to compare three or more independent groups. It can be used when the assumptions of normality and equal variances are not met for performing an ANOVA[163]. For more information on Kruskal-Wallis Test, refer to **Box 13**. Additionally, **Figure 19** shows a visual explanation of the Kruskal-Wallis algorithm, accompanied by the R-code used to test a feature across various groups and determine its significance.

Step 66: Perform Kruskal-Wallis Test on one feature

(User Input Required)

Begin by specifying the attribute for the Kruskal-Wallis (KW) test by entering its index number. In this tutorial, we opt for '`ATTRIBUTE_Sample_Area`'. Then, apply the KW test on a single feature (the first feature in the `` `uni_data` `` dataframe) across different sample areas using the `kruskal.test()` function. Note that the `` `uni_data` `` dataframe originates from the `` `cleaned_data` ``, which we chose as the `` `Imp_s` `` scaled table (see Step 24). Summarize the output into a one-row table using the `tidy()` function from the broom package as shown in the figure.

The steps for the Kruskal-Wallis test (steps 66 to 68) are structured similarly to the ANOVA steps (steps 54 to 57).

Step 67: Run Kruskal-Wallis Test for All Features

- Just like in ANOVA (step 55), perform the Kruskal-Wallis test for each metabolite across different sample areas. Then, tidy up the output for each feature into a table using the `tidy()` function.
- Combine these rows into a single dataframe containing features, their corresponding p-values, their BH-corrected p-values, and their significance status. Features with a BH-corrected p-value (`kruskall_out$p_BH < 0.05`) less than 0.05 are considered significant.

Step 68: Filter Significant Features

Display the count of significant and non-significant features. Filter out the names of significant features for further analysis.

Step 69: Visualize Kruskal-Wallis Results

Similar to ANOVA results, we first sort the `kruskall_out` dataframe results by p-value and visualize the significant features using `ggplot()`. This involves plotting log-transformed K-Statistic values on the x-axis against '`-log(p_BH)`' on the y-axis. To prevent clutter, limit the display to the names of the top 6 significant features.

Step 70: Visualize Top Significant Metabolites of Kruskal-Wallis Results

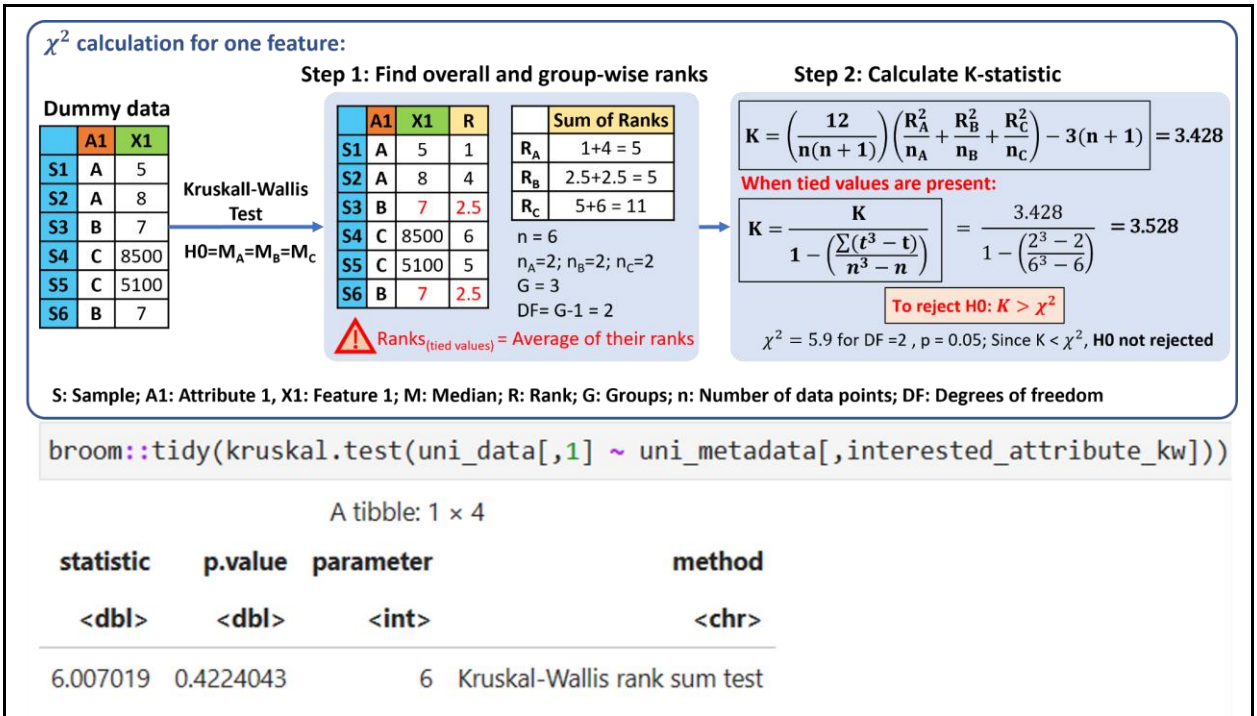
Generate boxplots for the top 4 significant metabolites to observe how their intensity levels differ across sampling sites. Extract these metabolites' data from the `uni_data` dataframe, which contains both feature intensities and metadata, and plot their intensities based on the sampling sites.

Step 71: Compare Results from ANOVA and Kruskal-Wallis

We also suggest comparing the significant outcomes from both ANOVA and Kruskal-Wallis tests. Features yielding high scores in both tests indicate that the null hypothesis is rejected by both ANOVA and Kruskal-Wallis. This suggests that these features show significant differences across groups (in our case, across different sample areas). This comparison can help prioritize the features for further analysis.

Box 13 - Kruskal-Wallis test

Although the Kruskal-Wallis test does not assume normality, it is expected that samples are random and independent and that the observations in each group come from populations with the same shape of distribution[163]. As an extension of the Mann–Whitney U test (which is used to compare only two groups), it compares the median ranks of the groups, which are calculated by combining the ranks of all the observations across all groups and then taking their average[164]. With this information, the K statistic can be calculated and compared to the chi-square distribution to accept or reject the null hypothesis (**Figure 19**). If the null hypothesis is rejected, the alternative hypothesis states that at least one group has a different median from the others.




```
broom::tidy(kruskal.test(uni_data[,1] ~ uni_metadata[,interested_attribute_kw]))
```

A tibble: 1 × 4

statistic	p.value	parameter	method
<dbl>	<dbl>	<int>	<chr>
6.007019	0.4224043	6	Kruskal-Wallis rank sum test

Figure 19: Kruskal-Wallis Test: The illustration provides a comprehensive view of the Kruskal-Wallis test algorithm. If the test results in rejecting the null hypothesis (with p < 0.05), it suggests that at least one group's median deviates significantly from the others. To complement the illustration, the corresponding code snippet from the protocol is presented. Echoing the approach with ANOVA, here the Kruskal-Wallis test is executed on an individual feature in relation to the metadata column that groups information, with our primary interest being the "Sample area".

3.4.3.2. Dunn's Post Hoc Test

The Dunn statistical test is a non-parametric alternative to the Tukey HSD post hoc test to make pairwise comparisons between multiple groups. The steps for Dunn's post hoc test (steps 72 to 75) are structured similarly to the Tukey HSD steps (steps 59 to 62). Refer to **Box 14** for more information on Dunn's post hoc test. **Figure 20** shows a visual representation for applying the Dunn test and its implementation in R.

Step 72: Perform Dunn Test for a Significant Feature
First, we select the first feature identified as significant in the KW test result, using `kw_sig_names` generated in step 68. From the KW output, we subset the data for this significant feature and conduct a Dunn test using `dunnTest()` function. The output is a comprehensive table providing an assessment of every possible pairwise group difference as shown in **Figure 20**.

When conducting a Dunn test on all significant features, consider specifying just one pair interaction to maintain simplicity. Similar to the Tukey HSD test, here we will focus on the results from comparisons between 'Mission Bay' and 'La Jolla Reefs' in the subsequent step.

Step 73: Perform Dunn Test for All Significant Features

(User Input Required)

Carry out a Dunn test for all the significant features identified in the Kruskal-Wallis test with BH correction for p-values. Then, filter the results for the specific interaction 'Mission Bay vs La Jolla Reefs'. To perform this, the user will be prompted to enter the index number corresponding to the desired comparison. This index number can be referenced from the table produced in the preceding step. Then, the Dunn Test result for those comparisons will be filtered for each feature showing the corrected p-values. The significance is assigned based on the corrected p-values (`dunn_output$P.adj < 0.05`) to identify the features that show a significant difference between these two sites.

Step 74: Count Significant Features

Determine how many features exhibit a significant difference between the chosen sites and how many do not.

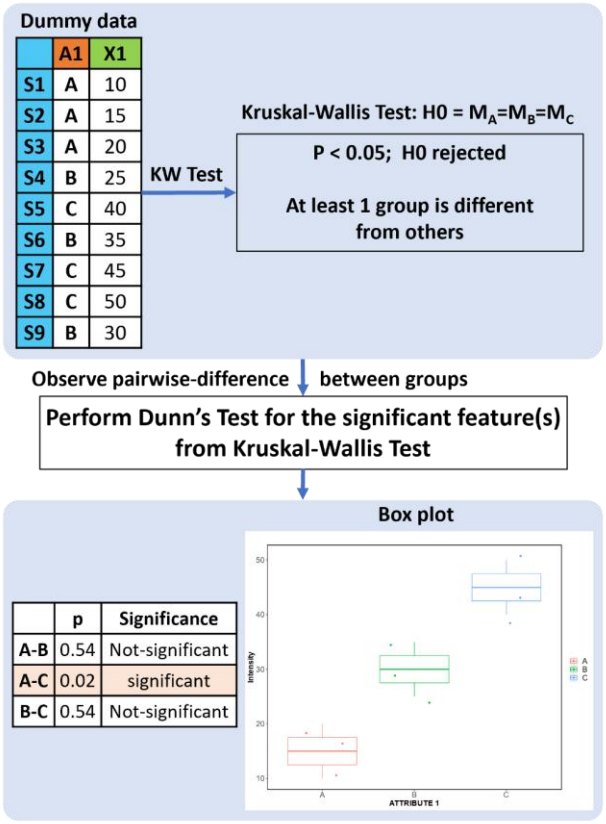
Step 75: Visualize Results with a Volcano Plot

Create a volcano plot with '`-log(p_BH)`' on the y-axis and the Z statistic on the x-axis. Display the names of the top findings on the plot to highlight the most significant differences between the chosen sites. Additionally, visualize the top 2 significant metabolites as boxplots from both extremes of the volcano plot (right and left tips) to clearly represent if the significant metabolite is upregulated or downregulated for the chosen sites.

For all the above tests, the respective significance values can be saved as a CSV table, and the plots can be saved in SVG, PDF, or PNG formats for further analysis or presentation.

Box 14 - Dunn test

> The Dunn statistical test is a non-parametric post-hoc test following Kruskal-Wallis test similar to the Tukey HSD post hoc test for ANOVA to make pairwise comparisons between multiple groups. Dunn's z-test approximation of the exact rank-sum test statistics is calculated with the mean rankings from the preceding Kruskal–Wallis test based on the differences in mean ranks for each group and, then, the p-value is calculated using a modified version of the BH correction to account for the type I error rate increase due to multiple comparisons[165] (**Figure 20**).



	A1	X1
S1	A	10
S2	A	15
S3	A	20
S4	B	25
S5	C	40
S6	B	35
S7	C	45
S8	C	50
S9	B	30

	p	Significance
A-B	0.54	Not-significant
A-C	0.02	significant
B-C	0.54	Not-significant

```
#Performing Dunn Test on the first significant feature of Kruskal Wallis output
dunn_first_sig_feature_model <- dunnTest(uni_data[,kw_sig_names[1]],
                                         as.factor(uni_metadata[,interested_attribute_kw]),
                                         method="bh")

data.frame(INDEX= 1:nrow(dunn_first_sig_feature_model$res),
           dunn_first_sig_feature_model$res)
```

A data.frame: 21 × 5

INDEX	Comparison	Z	P.unadj	P.adj
<int>	<chr>	<dbl>	<dbl>	<dbl>
1	La_Jolla Reefs - La_Jolla_Cove	1.93197175	5.336299e-02	1.018748e-01
2	La_Jolla Reefs - Mission_Bay	0.43199846	6.657425e-01	7.358207e-01
3	La_Jolla_Cove - Mission_Bay	-1.62650271	1.038428e-01	1.817248e-01
4	La_Jolla Reefs - Mission_Beach	0.99538701	3.195481e-01	4.793221e-01
5	La_Jolla_Cove - Mission_Beach	-0.95698460	3.385750e-01	4.740050e-01
6	Mission_Bay - Mission_Beach	0.64266175	5.204436e-01	6.429009e-01
7	La_Jolla Reefs - Pacific_Beach	0.36304436	7.165717e-01	7.524003e-01
8	La_Jolla_Cove - Pacific_Beach	-1.28102385	2.001853e-01	3.233762e-01
9	Mission_Bay - Pacific_Beach	0.05757532	9.540869e-01	9.540869e-01
10	Mission_Beach - Pacific_Beach	-0.44630672	6.553757e-01	7.646050e-01
11	La_Jolla Reefs - SIO_La_Jolla_Shores	5.59374261	2.222263e-08	1.166688e-07
12	La_Jolla_Cove - SIO_La_Jolla_Shores	2.60488633	9.190475e-03	1.930000e-02
13	Mission_Bay - SIO_La_Jolla_Shores	5.24101734	1.596937e-07	5.589280e-07
14	Mission_Beach - SIO_La_Jolla_Shores	3.98229276	6.825363e-05	1.791658e-04
15	Pacific_Beach - SIO_La_Jolla_Shores	4.00817764	6.118909e-05	1.835673e-04
16	La_Jolla Reefs - Torrey_Pines	8.32676723	8.307987e-17	1.744677e-15
17	La_Jolla_Cove - Torrey_Pines	3.69293308	2.216824e-04	5.172589e-04
18	Mission_Bay - Torrey_Pines	7.86494145	3.692701e-15	3.877336e-14
19	Mission_Beach - Torrey_Pines	5.60281453	2.108989e-08	1.476292e-07
20	Pacific_Beach - Torrey_Pines	5.31331432	1.076491e-07	4.521263e-07
21	SIO_La_Jolla_Shores - Torrey_Pines	0.79998875	4.237173e-01	5.561290e-01

Figure 20: Dunn Test for Post Hoc Analysis: This figure illustrates the Dunn test, a post hoc analysis following the Kruskal-Wallis test. After identifying significant features from the Kruskal-Wallis test, the next step is to conduct pairwise comparisons between groups. The accompanying code block demonstrates the execution of the Dunn test on the first significant feature obtained from the Kruskal-Wallis test, examining its relationship with various sample areas. The resulting display includes p-values for all potential 21 pairwise comparisons.

4. Example Study

Data Refinement and Annotation Insights

In the example data, we investigated the coastal environments along the San Diego coastline from Torrey Pines State Beach to Mission Bay, USA, during different dry and wet seasons. Refer to **Figure 21A** for a spatial map of the sampling locations. The presumption was that post-rain samples, influenced by runoff, would show increased pollutant levels. From FBMN analysis, we identified 5521 LC-MS/MS features, which decreased to 4384 after removing blanks. The library search against the GNPS spectral library via the FBMN workflow resulted in 92 annotated features out of the 4384 features, and an additional analog search putatively annotated 104 features. Expanding on this, we included additional data from October 2018, collected from the same sites (no-rain period). for our pipeline evaluation. The dataset contained 180 samples from seven different sites at three different time points (Dec 2017, Jan 2018, Oct 2018) and 2 PPL process blanks for each sample time. From this extended dataset, we identified 11217 features, with 260

GNPS library matches and 1991 analog matches. When focusing solely on December and January samples, the feature count surged to 10470, almost double the original count of 5521 features, and 240 GNPS library hits and 1624 analog hits.

To further expand our annotations, we used SIRIUS for in silico spectrum annotation on the extended dataset. We utilized the mgf file obtained from MZmine 3 and extended our SIRIUS analysis using tools like CANOPUS and CSI:FingerID. The SIRIUS result provided annotations for 8255 features, with annotations or compound names available for 5001 features. All 5001 of these features were further characterized by CSI:FingerID, which predicts molecular substructures and scores them based on the likelihood that the substructure belongs to the molecule. Leveraging the predictive capabilities of both SIRIUS and CSI:FingerID, we could infer the most probable molecular formulas. SIRIUS formula identifications were generated for 8885 features, with 5411 of these having an explained intensity greater than 80%, marking them as reliable formulas. For compound class predictions, CANOPUS provided annotations for 8583 features spanning various levels such as Kingdom, Superclass, Class, Subclass, and Level 5. On the other hand, the Natural Product Classifier (NPC) was used to determine if a compound is a natural product. These compound classes can be further explored in tools like Cytoscape for network visualization based on compound classes, or sub-setting of feature for subsequent statistics.

Impact of Sequential Data Cleanup

Contaminant features, especially those exceeding 30% peak area relative to the sample average, were flagged and removed, leaving us with 9,092 features. Our dataset showed 32% missing values out of 1,636,560 total entries, which were imputed between 1 and the lowest feature value (892). Petras *et al*. found significant organic matter chemotype shifts between December 2017 and January 2018 samples, correlating with January's heavy rainfall[14]. Our extended dataset confirmed this, with a PCoA analysis revealing clear sample groupings by the sampling month as shown in **Figure 21B**. Post-blank removal intensified these groupings. Prior to data cleanup, no dispersion effect was apparent ($p > 0.05$), and PERMANOVA attributed 31% of the variance to sampling months. After removing blanks, however, a dispersion effect emerged. This dispersion effect and explained variance in PERMANOVA are likely due to the removal of background features, thus reflecting the true water sample chemotypes for each month. Upon examining the PCoA after imputation, individual clusters appeared closer together, though January samples exhibited some dispersion. This spread within January samples became more pronounced after normalization and scaling.

Multivariate Analysis: Diving into Site-Specific Variations

Using PERMANOVA on the scaled-imputed data, we identified a significant clustering by months, attributing 34% of variance to the sampling time ($P < 0.05$, Adonis $R2 = 0.34$). Sample locations, however, explained only 7% of the variance. Upon deeper exploration at the metabolic profiles across these sampling locations, January's variance was more prominent in Mission Bay, especially post-rainfall, due to its nutrient-rich status, potentially from increased runoff through the San Diego River. This distinction is evident in the PCoA plot in **Figure 21B**. Our data showed

Mission Bay's pre-rainfall samples were similar to other sites, but post-rain samples in January diverged — a pattern absent in December 2017 and October 2018 samples. We could also observe some clear patterns in the heatmap depicted in **Figure 21C**. Color transitions from blue (0 intensity) to red (1 intensity) highlight feature intensity variations. Many features were found in higher intensities in October samples compared to December and January samples. Mission Bay samples from January (in red) and a subset from Torrey Pines (in blue) displayed increased feature intensities. This aligns well with our initial hypothesis. Alongside this, we performed a random forest classification considering sampling sites.

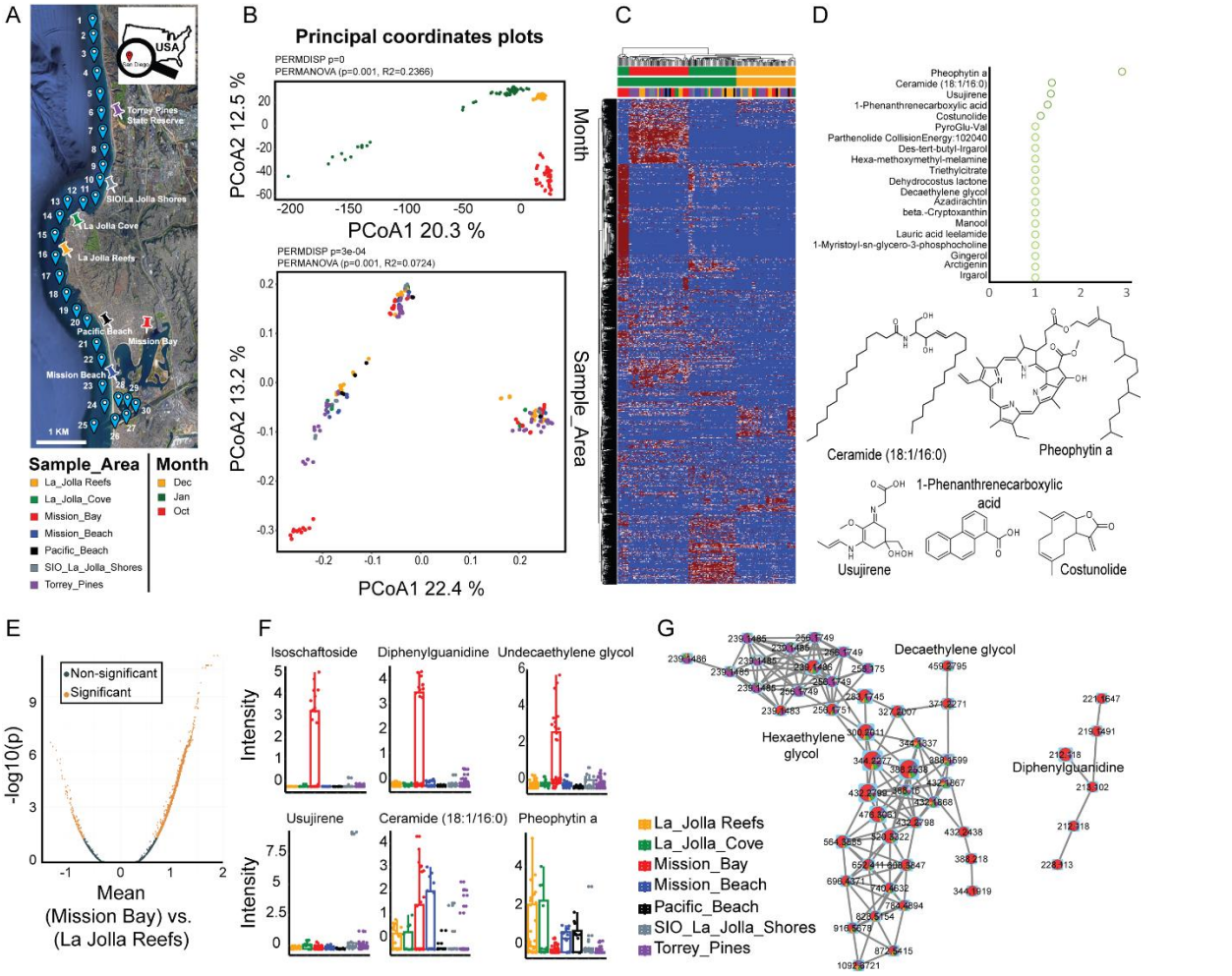


Figure 21: Anticipated results: A) Spatial map pinpointing sampling sites; B) Principal coordinate plots delineating differences by sampling month and location; C) Heatmap displaying scaled feature intensities; D) Top 20 annotated drivers for temporal changes identified via Random Forest, with structures of the top 5 metabolites shown; E) Volcano plot illustrating the Tukey test comparison between Mission Bay and La Jolla Reefs samples, with features deemed significant in the ANOVA Sample area-based test used for this post-hoc analysis; F) Box plots illustrating feature intensities across various sampling locations. The top row presents the foremost 3 annotated significant outcomes post-Tukey test, accompanied by their molecular structures. Conversely, the second row highlights the top 3 significant outputs as identified by Random Forest; G) Molecular Networks of significant features (diphenylguanidine and polyethylene glycols, highlight related molecules with similar spatial patterns as indicated through the pie charts on top of the network nodes.

Random Forest Exploration: Prioritizing Key Drivers

Utilizing a Random Forest model with 500 trees and 500 permutations, we attained a 68.3% prediction accuracy for the samples. By location, accuracy ranged from 87.5% (Torrey Pines) to 16.7% (Pacific Beach). The confusion matrix in **Table 4** provides insights into these results, revealing that misclassifications were often between neighboring sites, likely due to the close 300-meter spacing between the sampling locations. Our model highlighted 438 significant features (based on 'Mean Decrease Accuracy p value'). Of these, seven matched GNPS libraries and 96 were analog hits. Examining the violin plot results of RF, top features, like those with library IDs 91372 and 90597 (both sharing the same analog name), were mainly concentrated in Mission Bay and La Jolla Reefs. These concentrations began low at Torrey Pines, peaked at Cove and Reef, and saw another spike in Mission Bay. Similar patterns emerged for features like theaflavin digallatae (ID 91133). Some features, such as IDs 33200 and 53617, were notably elevated in Mission Bay alone. Certain compounds from previously reported research, such as *m/z* 1129.3145 (analog name: benzyl-tetradecyl-dimethylammonium) specific to January samples, were also detected in our study, but their significance was marginal ($p = 0.08$) and was predominantly seen in Torrey Pines. Several compounds reported in the original study such as irgarol, recognized for their pollution potential and unique spatial patterns, were also explored in our dataset. **Figure 21D** visualizes the top 20 annotated drivers for site-specific changes as identified via Random Forest, highlighting the structures of the top 5 metabolites. In summary, our extended data set enhances the Random Forest analysis, offering a detailed understanding of chemotype differences across coastal areas and reaffirming the conclusions of the original study.

Table 4: Confusion Matrix of Random Forest Classification

The confusion matrix shows how many samples from each group were correctly predicted. Taking the first row as an example: out of 36 samples from La Jolla Reefs, 25 were accurately identified. The remaining samples were misclassified as follows: 1 as Mission Bay, 1 as Mission Beach, 5 as Pacific Beach, and 4 as SIO La Jolla Shores. The column labeled 'pct.correct' represents the percentage of samples that were correctly classified for a given group. The columns 'LCI 0.95' and 'UCI 0.95' denote the lower and upper bounds of the 95% confidence interval for each group, respectively. The 'overall' row at the bottom indicates the model's total prediction accuracy, which stands at 68.3% for this dataset.

	La Jolla Reefs	La Jolla Cove	Mission Bay	Mission Beach	Pacific Beach	SIO La Jolla Shores	Torrey Pines	pct.correct	LCI 0.95	UCI 0.95
La Jolla Reefs	25	0	1	1	5	4	0	69.4	51.89	83.7
La Jolla Cove	0	10	0	0	0	2	0	83.3	51.59	97.9
Mission Bay	4	0	23	7	2	0	0	63.9	46.22	79.2
Mission Beach	0	0	0	15	3	0	0	83.3	58.58	96.4
Pacific Beach	6	0	1	3	2	0	0	16.7	2.09	48.4
SIO La Jolla Shores	2	0	0	1	0	6	9	33.3	13.34	59
Torrey Pines	0	0	0	0	0	6	42	87.5	74.75	95.3
Overall	NA	NA	NA	NA	NA	NA	NA	68.3	61	75.1

Univariate Analysis Insights

In our univariate analysis of 9092 features, both ANOVA and the Kruskal-Wallis test were utilized. However, the Kruskal-Wallis test was considered more apt due to the non-parametric nature of our dataset. The Kruskal-Wallis test highlighted 1258 significant features, including irgarol, an antifouling agent used on boats. Conversely, ANOVA pinpointed 1554 significant features, with many features having a pronounced abundance in Mission Bay compared to other sites. Notably, one of the features corresponded to hexaethylene glycol from the NIST14 database and several features matched to nanoethylene glycols, which fall under the PEGs (Polyethylene glycols) category. Another notable find was an analog match to sporidesmolide 2, previously identified in the base study.

Building on the ANOVA results, Tukey's HSD test was used to highlight pairwise differences. Given the pronounced abundance of many features in Mission Bay, we compared it with La Jolla Reefs for further insights. The significant and non-significant features from this test are visualized in the volcano plot in **Figure 21E**. Notably, compounds like 1,2-diphenylguanidine (used in metal detection and rubber vulcanization) and nanoethylene glycol were significantly higher in Mission Bay. In contrast, La Jolla Reefs had a higher presence of the natural product 'pheophytin a' at various retention times (RT 11.312, 11.022). The top row in **Figure 21F** displays the intensities of the top three annotated results from the Tukey test across the sampling locations using box plots, each paired with its corresponding molecular structure. Interestingly, 'pheophytin a' was also more abundant in La Jolla Reefs in subsequent Dunn Tests post-Kruskal-Wallis. Furthermore, irgarol was consistently found to be more abundant in Mission Bay in both tests.

These findings align with and reinforce the initial observations, validating the robustness of our analytical workflow.

Integration of Molecular Networking Results

After the statistical analysis of the FBMN results and prioritization of features that drive the chemical differences between the sampling sites, we further investigate related compounds, through the molecular networks. **Figure 21G** shows the networks of diphenylguanidine and polyethylene glycols, indicating that many of the structurally related features of those compounds show similar spatial distribution, with the highest abundance in Mission Bay, as indicated through the pie charts on top of the network nodes. These results show nicely how the statistical prioritization and further structure-based (in our case, based on MS/MS similarity) can work hand in hand to structure the observed chemical space. Besides investigating the networks after the statistical interrogation, one can also make use of the scores obtained from the different tests and visualize those in the network. For example, the fold change and p-values from the univariate analysis or mean decreased accuracies form the supervised multivariate analysis can be imported as new attribute to the networks with tools such as Cytoscape to combine visual and statistical prioritization directly in the network.

5. Conclusion

In this protocol, we provide a comprehensive data clean-up and statistics pipeline for the analysis of non-targeted metabolomics data. Our protocol spans from initial data conversion, blank removal, imputation, and normalization/scaling to uni- and multivariate statistics and data interpretation. While our outlined workflow is as detailed and structured as possible, which should provide a comprehensive analysis solution for many biological questions, it is important to point out that there is not a universal solution that fits every scenario. We emphasize the importance of transparency in reporting details on every step of the metabolomics pipeline, such as providing the specific normalization methods, explaining the distance metrics in multivariate analysis, or specifying parameters like the number of trees in a Random Forest model. Furthermore, in relation to our case study, the sharing of feature detection and annotation settings and batch files further augments reproducibility. Together, with open data deposition, the above steps ensure both transparency and reproducibility of metabolomics experiments.

We would also like to stress again that cataloging and identifying statistically significant metabolites is just the beginning. To fully understand the relationships between metabolites and the underlying biological processes, additional experiments and orthogonal verification are typically required. Once the statistical results are studied, techniques such as pathway enrichment analyses can illuminate the multifaceted relationships between metabolites and the biological processes they are entwined with. When specific compounds are of particular interest, targeted metabolomics stands as a powerful next step.

In summary, we anticipate that our Hitchhicker's Guide to statistical analysis of FBMN results will provide both a theoretical and practical resource for scientists working with non-targeted metabolomics data. For novices in the field, the scripts, app and detailed step-to-step protocol provide a starting point with a set of statistical analysis solutions for many biological questions, whereas experts may accelerate parts of their statistical workflows.

6. References

1. Vailati-Riboni, M., Palombo, V. & Loor, J. J. What Are Omics Sciences? in *Periparturient Diseases of Dairy Cows: A Systems Biology Approach* (ed. Ametaj, B. N.) 1–7 (Springer International Publishing, 2017). doi:10.1007/978-3-319-43033-1_1.
2. Patti, G. J., Yanes, O. & Siuzdak, G. Metabolomics: the apogee of the omics trilogy. *Nat. Rev. Mol. Cell Biol.* **13**, 263–269 (2012).
3. Dayalan, S., Xia, J., Spicer, R. A., Salek, R. & Roessner, U. Metabolome Analysis. in *Encyclopedia of Bioinformatics and Computational Biology* (eds. Ranganathan, S., Gribskov, M., Nakai, K. & Schönbach, C.) 396–409 (Academic Press, 2019). doi:10.1016/B978-0-12-809633-8.20251-3.
4. Tolstikov, V., Moser, A. J., Sarangarajan, R., Narain, N. R. & Kiebish, M. A. Current Status of Metabolomic Biomarker Discovery: Impact of Study Design and Demographic Characteristics. *Metabolites* **10**, 224 (2020).
5. de Jonge, N. F. *et al.* Good practices and recommendations for using and benchmarking computational metabolomics metabolite annotation tools. *Metabolomics* **18**, 103 (2022).

6. Nothias, L.-F. *et al.* Feature-based molecular networking in the GNPS analysis environment. *Nat. Methods* **17**, 905–908 (2020).
7. Wang, M. *et al.* Sharing and community curation of mass spectrometry data with Global Natural Products Social Molecular Networking. *Nat. Biotechnol.* **34**, 828–837 (2016).
8. Ottosson, F. *et al.* Effects of Long-Term Storage on the Biobanked Neonatal Dried Blood Spot Metabolome. *J. Am. Soc. Mass Spectrom.* **34**, 685–694 (2023).
9. Dantas Machado, A. C. *et al.* Portosystemic shunt placement reveals blood signatures for the development of hepatic encephalopathy through mass spectrometry. *Nat. Commun.* **14**, 5303 (2023).
10. Xie, H.-F. *et al.* Feature-based molecular networking analysis of the metabolites produced by in vitro solid-state fermentation reveals pathways for the bioconversion of epigallocatechin gallate. *J. Agric. Food Chem.* **68**, 7995–8007 (2020).
11. Berlanga-Clavero, M. V. *et al.* Bacillus subtilis biofilm matrix components target seed oil bodies to promote growth and anti-fungal resistance in melon. *Nat. Microbiol.* **7**, 1001–1015 (2022).
12. Raheem, D. J., Tawfike, A. F., Abdelmohsen, U. R., Edrada-Ebel, R. & Fitzsimmons-Thoss, V. Application of metabolomics and molecular networking in investigating the chemical profile and antitrypanosomal activity of British bluebells (Hyacinthoides non-scripta). *Sci. Rep.* **9**, 2547 (2019).
13. Pendergraft, M. A. *et al.* Bacterial and Chemical Evidence of Coastal Water Pollution from the Tijuana River in Sea Spray Aerosol. *Environ. Sci. Technol.* **57**, 4071–4081 (2023).
14. Petras, D. *et al.* Non-targeted tandem mass spectrometry enables the visualization of organic matter chemotype shifts in coastal seawater. *Chemosphere* **271**, 129450 (2021).
15. Stincone, P. *et al.* Evaluation of Data-Dependent MS/MS Acquisition Parameters for Non-Targeted Metabolomics and Molecular Networking of Environmental Samples: Focus on the Q Exactive Platform. *Anal. Chem.* **95**, 12673–12682 (2023).
16. Wegley Kelly, L. *et al.* Distinguishing the molecular diversity, nutrient content, and energetic potential of exometabolomes produced by macroalgae and reef-building corals. *Proc. Natl. Acad. Sci.* **119**, e2110283119 (2022).
17. Mannochio-Russo, H. *et al.* Microbiomes and metabolomes of dominant coral reef primary producers illustrate a potential role for immunolipids in marine symbioses. *Commun. Biol.* **6**, 896 (2023).
18. Shaffer, J. P. *et al.* Standardized multi-omics of Earth's microbiomes reveals microbial and metabolite diversity. *Nat. Microbiol.* **7**, 2128–2150 (2022).
19. Molina-Santiago, C. *et al.* Chemical interplay and complementary adaptative strategies toggle bacterial antagonism and co-existence. *Cell Rep.* **36**, (2021).
20. Reher, R. *et al.* Native metabolomics identifies the rivulariapeptolide family of protease inhibitors. *Nat. Commun.* **13**, 4619 (2022).
21. Aron, A. T. *et al.* Native mass spectrometry-based metabolomics identifies metal-binding compounds. *Nat. Chem.* **14**, 100–109 (2022).
22. Behnsen, J. *et al.* Siderophore-mediated zinc acquisition enhances enterobacterial colonization of the inflamed gut. *Nat. Commun.* **12**, 7016 (2021).
23. Pang, Z. *et al.* MetaboAnalyst 5.0: narrowing the gap between raw spectra and functional insights. *Nucleic Acids Res.* **49**, W388–W396 (2021).

24. Pang, Z. *et al.* Using MetaboAnalyst 5.0 for LC–HRMS spectra processing, multi-omics integration and covariate adjustment of global metabolomics data. *Nat. Protoc.* **17**, 1735–1761 (2022).
25. Cajka, T. & Fiehn, O. Toward Merging Untargeted and Targeted Methods in Mass Spectrometry-Based Metabolomics and Lipidomics. *Anal. Chem.* **88**, 524–545 (2016).
26. Alder, L., Greulich, K., Kempe, G. & Vieth, B. Residue analysis of 500 high priority pesticides: Better by GC–MS or LC–MS/MS? *Mass Spectrom. Rev.* **25**, 838–865 (2006).
27. Díaz-Cruz, M. S., López de Alda, M. J., López, R. & Barceló, D. Determination of estrogens and progestogens by mass spectrometric techniques (GC/MS, LC/MS and LC/MS/MS). *J. Mass Spectrom.* **38**, 917–923 (2003).
28. Michely, J. A., Helfer, A. G., Brandt, S. D., Meyer, M. R. & Maurer, H. H. Metabolism of the new psychoactive substances N,N-diallyltryptamine (DALT) and 5-methoxy-DALT and their detectability in urine by GC–MS, LC–MSn, and LC–HR–MS–MS. *Anal. Bioanal. Chem.* **407**, 7831–7842 (2015).
29. Di Masi, S. *et al.* HPLC-MS/MS method applied to an untargeted metabolomics approach for the diagnosis of “olive quick decline syndrome”. *Anal. Bioanal. Chem.* **414**, 465–473 (2022).
30. Reveglia, P. *et al.* Untargeted and Targeted LC-MS/MS Based Metabolomics Study on In Vitro Culture of Phaeoacremonium Species. *J. Fungi* **8**, 55 (2022).
31. Baig, F., Pechlaner, R. & Mayr, M. Caveats of Untargeted Metabolomics for Biomarker Discovery∗. *J. Am. Coll. Cardiol.* **68**, 1294–1296 (2016).
32. Xiao, J. F., Zhou, B. & Ressom, H. W. Metabolite identification and quantitation in LC-MS/MS-based metabolomics. *TrAC Trends Anal. Chem.* **32**, 1–14 (2012).
33. Blaženović, I. *et al.* Comprehensive comparison of in silico MS/MS fragmentation tools of the CASMI contest: database boosting is needed to achieve 93% accuracy. *J. Cheminformatics* **9**, 32 (2017).
34. Blaženović, I., Kind, T., Ji, J. & Fiehn, O. Software Tools and Approaches for Compound Identification of LC-MS/MS Data in Metabolomics. *Metabolites* **8**, 31 (2018).
35. Dührkop, K., Shen, H., Meusel, M., Rousu, J. & Böcker, S. Searching molecular structure databases with tandem mass spectra using CSI:FingerID. *Proc. Natl. Acad. Sci.* **112**, 12580–12585 (2015).
36. Böcker, S., Letzel, M. C., Lipták, Z. & Pervukhin, A. SIRIUS: decomposing isotope patterns for metabolite identification†. *Bioinformatics* **25**, 218–224 (2009).
37. Stravs, M. A., Dührkop, K., Böcker, S. & Zamboni, N. MSNovelist: de novo structure generation from mass spectra. *Nat. Methods* **19**, 865–870 (2022).
38. Aron, A. T. *et al.* Reproducible molecular networking of untargeted mass spectrometry data using GNPS. *Nat. Protoc.* **15**, 1954–1991 (2020).
39. Schmid, R. *et al.* Ion identity molecular networking for mass spectrometry-based metabolomics in the GNPS environment. *Nat. Commun.* **12**, 3832 (2021).
40. Dunn, W. B. *et al.* Procedures for large-scale metabolic profiling of serum and plasma using gas chromatography and liquid chromatography coupled to mass spectrometry. *Nat. Protoc.* **6**, 1060–1083 (2011).
41. Silva, A. M., Cordeiro-da-Silva, A. & Coombs, G. H. Metabolic Variation during Development in Culture of Leishmania donovani Promastigotes. *PLoS Negl. Trop. Dis.* **5**, e1451 (2011).

42. Martínez-Sena, T. *et al.* Monitoring of system conditioning after blank injections in untargeted UPLC-MS metabolomic analysis. *Sci. Rep.* **9**, 9822 (2019).
43. Steffen Heuckeroth, Tito Damiani, Aleksandr Smirnov, Olena Mokshyna, Corinna Brungs, Ansgar Korf4, Joshua David Smith, Paolo Stincone, Nicola Dreolin, Louis-Félix Nothias, Tuulia Hyötyläinen, Matej Orešič, Uwe Karst, Pieter C. Dorrestein, Daniel Petras, Xiuxia Du, Justin J.J. van der Hooft, Robin Schmid, Tomáš Pluskal. Mass spectrometry data processing in MZmine 3: feature detection and annotation. *Nat. Protoc.* **Under Review**, (2023).
44. Raynie, D. The Vital Role of Blanks in Sample Preparation. *LCGC N. Am.* **36**, 494–497 (2018).
45. Kessner, D., Chambers, M., Burke, R., Agus, D. & Mallick, P. ProteoWizard: open source software for rapid proteomics tools development. *Bioinformatics* **24**, 2534–2536 (2008).
46. Hulstaert, N. *et al.* ThermoRawFileParser: Modular, Scalable, and Cross-Platform RAW File Conversion. *J. Proteome Res.* **19**, 537–542 (2020).
47. Smith, C. A., Want, E. J., O'Maille, G., Abagyan, R. & Siuzdak, G. XCMS: Processing Mass Spectrometry Data for Metabolite Profiling Using Nonlinear Peak Alignment, Matching, and Identification. *Anal. Chem.* **78**, 779–787 (2006).
48. Kuhl, C., Tautenhahn, R., Böttcher, C., Larson, T. R. & Neumann, S. CAMERA: An Integrated Strategy for Compound Spectra Extraction and Annotation of Liquid Chromatography/Mass Spectrometry Data Sets. *Anal. Chem.* **84**, 283–289 (2012).
49. Schmid, R. *et al.* Integrative analysis of multimodal mass spectrometry data in MZmine 3. *Nat. Biotechnol.* **41**, 447–449 (2023).
50. Tsugawa, H. *et al.* A lipidome atlas in MS-DIAL 4. *Nat. Biotechnol.* **38**, 1159–1163 (2020).
51. Pfeuffer, J. *et al.* OpenMS – A platform for reproducible analysis of mass spectrometry data. *J. Biotechnol.* **261**, 142–148 (2017).
52. Gloaguen, Y., Kirwan, J. A. & Beule, D. Deep Learning-Assisted Peak Curation for Large-Scale LC-MS Metabolomics. *Anal. Chem.* **94**, 4930–4937 (2022).
53. Chetnik, K., Petrick, L. & Pandey, G. MetaClean: a machine learning-based classifier for reduced false positive peak detection in untargeted LC–MS metabolomics data. *Metabolomics* **16**, 117 (2020).
54. El Abiead, Y., Milford, M., Salek, R. M. & Koellensperger, G. mzRAPP: a tool for reliability assessment of data pre-processing in non-targeted metabolomics. *Bioinformatics* **37**, 3678–3680 (2021).
55. Tautenhahn, R., Böttcher, C. & Neumann, S. Highly sensitive feature detection for high resolution LC/MS. *BMC Bioinformatics* **9**, 504 (2008).
56. Dührkop, K. *et al.* SIRIUS 4: a rapid tool for turning tandem mass spectra into metabolite structure information. *Nat. Methods* **16**, 299–302 (2019).
57. Sumner, L. W. *et al.* Proposed minimum reporting standards for chemical analysis. *Metabolomics* **3**, 211–221 (2007).
58. Liu, L.-L. *et al.* Molecular networking-based for the target discovery of potent antiproliferative polycyclic macrolactam ansamycins from Streptomyces cacaoi subsp. asoensis. *Org. Chem. Front.* **7**, 4008–4018 (2020).
59. Sedio, B. E., Boya P., C. A. & Rojas Echeverri, J. C. A protocol for high-throughput, untargeted forest community metabolomics using mass spectrometry molecular networks. *Appl. Plant Sci.* **6**, e1033 (2018).

60. Quinn, R. A. *et al.* Molecular Networking As a Drug Discovery, Drug Metabolism, and Precision Medicine Strategy. *Trends Pharmacol. Sci.* **38**, 143–154 (2017).
61. Pluskal, T., Castillo, S., Villar-Briones, A. & Orešič, M. MZmine 2: Modular framework for processing, visualizing, and analyzing mass spectrometry-based molecular profile data. *BMC Bioinformatics* **11**, 395 (2010).
62. Djoumbou Feunang, Y. *et al.* ClassyFire: automated chemical classification with a comprehensive, computable taxonomy. *J. Cheminformatics* **8**, 61 (2016).
63. Kim, H. W. *et al.* NPClassifier: A Deep Neural Network-Based Structural Classification Tool for Natural Products. *J. Nat. Prod.* **84**, 2795–2807 (2021).
64. Dührkop, K. *et al.* Systematic classification of unknown metabolites using high-resolution fragmentation mass spectra. *Nat. Biotechnol.* **39**, 462–471 (2021).
65. Bolyen, E. *et al.* Reproducible, interactive, scalable and extensible microbiome data science using QIIME 2. *Nat. Biotechnol.* **37**, 852–857 (2019).
66. Davidson, R. L., Weber, R. J. M., Liu, H., Sharma-Oates, A. & Viant, M. R. Galaxy-M: a Galaxy workflow for processing and analyzing direct infusion and liquid chromatography mass spectrometry-based metabolomics data. *GigaScience* **5**, 10 (2016).
67. Giacomoni, F. *et al.* Workflow4Metabolomics: a collaborative research infrastructure for computational metabolomics. *Bioinformatics* **31**, 1493–1495 (2015).
68. Kontou, E. E. *et al.* UmetaFlow: an untargeted metabolomics workflow for high-throughput data processing and analysis. *J. Cheminformatics* **15**, 52 (2023).
69. Rohart, F., Gautier, B., Singh, A. & Lê Cao, K.-A. mixOmics: An R package for 'omics feature selection and multiple data integration. *PLoS Comput. Biol.* **13**, e1005752 (2017).
70. Chong, J. & Xia, J. MetaboAnalystR: an R package for flexible and reproducible analysis of metabolomics data. *Bioinformatics* **34**, 4313–4314 (2018).
71. Tiffany, C. R. & Bäumler, A. J. omu, a Metabolomics count data analysis tool for intuitive figures and convenient metadata collection. *Microbiol. Resour. Announc.* **8**, 10.1128/mra. 00129-19 (2019).
72. Han, X. & Liang, L. metabolomicsR: a streamlined workflow to analyze metabolomic data in R. *Bioinforma. Adv.* **2**, vbac067 (2022).
73. Fernández-Albert, F., Llorach, R., Andrés-Lacueva, C. & Perera, A. An R package to analyse LC/MS metabolomic data: MAIT (Metabolite Automatic Identification Toolkit). *Bioinformatics* **30**, 1937–1939 (2014).
74. Kohler, D. *et al.* MSstats Version 4.0: Statistical Analyses of Quantitative Mass Spectrometry-Based Proteomic Experiments with Chromatography-Based Quantification at Scale. *J. Proteome Res.* **22**, 1466–1482 (2023).
75. Riquelme, G., Zabalegui, N., Marchi, P., Jones, C. M. & Monge, M. E. A Python-Based Pipeline for Preprocessing LC–MS Data for Untargeted Metabolomics Workflows. *Metabolites* **10**, 416 (2020).
76. Di Guida, R. *et al.* Non-targeted UHPLC-MS metabolomic data processing methods: a comparative investigation of normalisation, missing value imputation, transformation and scaling. *Metabolomics* **12**, 93 (2016).
77. Watrous, J. *et al.* Mass spectral molecular networking of living microbial colonies. *Proc. Natl. Acad. Sci.* **109**, E1743–E1752 (2012).

78. Hoffmann, M. A. *et al.* Assigning confidence to structural annotations from mass spectra with COSMIC. 2021.03.18.435634 Preprint at https://doi.org/10.1101/2021.03.18.435634 (2021).
79. Rinker, T. *et al.* pacman: Package Management Tool. (2019).
80. Wickham, H. *et al.* Welcome to the Tidyverse. *J. Open Source Softw.* **4**, 1686 (2019).
81. Kluyver, T., Angerer, P. & Schulz, J. IRdisplay: 'Jupyter' Display Machinery. (2022).
82. Cacciatore, S., Luchinat, C. & Tenori, L. Knowledge discovery by accuracy maximization. *Proc. Natl. Acad. Sci.* **111**, 5117–5122 (2014).
83. Kassambara, A. & Mundt, F. Extract and Visualize the Results of Multivariate Data Analyses [R package factoextra version 1.0.7]. in (2020).
84. Oksanen, J. *et al.* vegan: Community Ecology Package. (2022).
85. Gu, Z. Complex heatmap visualization. *iMeta* **1**, e43 (2022).
86. Galili, T. dendextend: an R package for visualizing, adjusting and comparing trees of hierarchical clustering. *Bioinforma. Oxf. Engl.* **31**, 3718–3720 (2015).
87. Charrad, M., Ghazzali, N., Boiteau, V. & Niknafs, A. NbClust: An R Package for Determining the Relevant Number of Clusters in a Data Set. *J. Stat. Softw.* **61**, 1–36 (2014).
88. Archer, E. rfPermute: Estimate Permutation p-Values for Random Forest Importance Metrics. (2023).
89. Ogle, D. H., Doll, J. C., Wheeler, A. P. & dunnTest()), A. D. (Provided base functionality of. FSA: Simple Fisheries Stock Assessment Methods. (2023).
90. Bengtsson, H. *et al.* matrixStats: Functions that Apply to Rows and Columns of Matrices (and to Vectors). (2023).
91. Xiao [aut, N., cre, Cook, J., Jégousse, C. & Li, M. ggsci: Scientific Journal and Sci-Fi Themed Color Palettes for 'ggplot2'. (2023).
92. Wilke, C. O. cowplot: Streamlined Plot Theme and Plot Annotations for 'ggplot2'. (2020).
93. Wickham, H. *et al.* svglite: An 'SVG' Graphics Device. (2023).
94. Liu, Q. *et al.* Addressing the batch effect issue for LC/MS metabolomics data in data preprocessing. *Sci. Rep.* **10**, 13856 (2020).
95. Yue, Y., Bao, X., Jiang, J. & Li, J. Evaluation and correction of injection order effects in LC-MS/MS based targeted metabolomics. *J. Chromatogr. B* **1212**, 123513 (2022).
96. Livera, A. M. D. *et al.* Statistical Methods for Handling Unwanted Variation in Metabolomics Data. *Anal. Chem.* **87**, 3606–3615 (2015).
97. Reese, S. E. *et al.* A new statistic for identifying batch effects in high-throughput genomic data that uses guided principal component analysis. *Bioinformatics* **29**, 2877–2883 (2013).
98. Burton, L. *et al.* Instrumental and experimental effects in LC–MS-based metabolomics. *J. Chromatogr. B* **871**, 227–235 (2008).
99. Gregori, J. *et al.* Batch effects correction improves the sensitivity of significance tests in spectral counting-based comparative discovery proteomics. *J. Proteomics* **75**, 3938–3951 (2012).
100. Thonusin, C. *et al.* Evaluation of intensity drift correction strategies using MetaboDrift, a normalization tool for multi-batch metabolomics data. *J. Chromatogr. A* **1523**, 265–274 (2017).
101. Johnson, W. E., Li, C. & Rabinovic, A. Adjusting batch effects in microarray expression data using empirical Bayes methods. *Biostatistics* **8**, 118–127 (2007).

102. Deng, K. *et al.* WaveICA: A novel algorithm to remove batch effects for large-scale untargeted metabolomics data based on wavelet analysis. *Anal. Chim. Acta* **1061**, 60–69 (2019).
103. Wehrens, R. *et al.* Improved batch correction in untargeted MS-based metabolomics. *Metabolomics* **12**, 88 (2016).
104. Kuligowski, J., Sánchez-Illana, Á., Sanjuán-Herráez, D., Vento, M. & Quintás, G. Intra-batch effect correction in liquid chromatography-mass spectrometry using quality control samples and support vector regression (QC-SVRC). *The Analyst* **140**, 7810–7817 (2015).
105. Luan, H., Ji, F., Chen, Y. & Cai, Z. statTarget: A streamlined tool for signal drift correction and interpretations of quantitative mass spectrometry-based omics data. *Anal. Chim. Acta* **1036**, 66–72 (2018).
106. Rong, Z. *et al.* NormAE: Deep Adversarial Learning Model to Remove Batch Effects in Liquid Chromatography Mass Spectrometry-Based Metabolomics Data. *Anal. Chem.* **92**, 5082–5090 (2020).
107. Dmitrenko, A., Reid, M. & Zamboni, N. Regularized adversarial learning for normalization of multi-batch untargeted metabolomics data. *Bioinformatics* **39**, btad096 (2023).
108. Tokareva, A. O. *et al.* Normalization methods for reducing interbatch effect without quality control samples in liquid chromatography-mass spectrometry-based studies. *Anal. Bioanal. Chem.* **413**, 3479–3486 (2021).
109. Cleary, J. L., Luu, G. T., Pierce, E. C., Dutton, R. J. & Sanchez, L. M. BLANKA: an Algorithm for Blank Subtraction in Mass Spectrometry of Complex Biological Samples. *J. Am. Soc. Mass Spectrom.* **30**, 1426–1434 (2019).
110. Lawson, T. N. *et al.* msPurity: Automated Evaluation of Precursor Ion Purity for Mass Spectrometry-Based Fragmentation in Metabolomics. *Anal. Chem.* **89**, 2432–2439 (2017).
111. Schiffman, C. *et al.* Data-adaptive pipeline for filtering and normalizing metabolomics data. 387365 Preprint at https://doi.org/10.1101/387365 (2018).
112. Carobene, A., Braga, F., Roraas, T., Sandberg, S. & Bartlett, W. A. A systematic review of data on biological variation for alanine aminotransferase, aspartate aminotransferase and γ-glutamyl transferase. *Clin. Chem. Lab. Med. CCLM* **51**, 1997–2007 (2013).
113. Schiffman, C. *et al.* Filtering procedures for untargeted LC-MS metabolomics data. *BMC Bioinformatics* **20**, 334 (2019).
114. Broadhurst, D. *et al.* Guidelines and considerations for the use of system suitability and quality control samples in mass spectrometry assays applied in untargeted clinical metabolomic studies. *Metabolomics* **14**, 72 (2018).
115. Wei, R. *et al.* Missing Value Imputation Approach for Mass Spectrometry-based Metabolomics Data. *Sci. Rep.* **8**, 663 (2018).
116. Do, K. T. *et al.* Characterization of missing values in untargeted MS-based metabolomics data and evaluation of missing data handling strategies. *Metabolomics* **14**, 128 (2018).
117. Gorrochategui, E., Jaumot, J., Lacorte, S. & Tauler, R. Data analysis strategies for targeted and untargeted LC-MS metabolomic studies: Overview and workflow. *TrAC Trends Anal. Chem.* **82**, 425–442 (2016).
118. Wulff, J. E. & Mitchell, M. W. A Comparison of Various Normalization Methods for LC/MS Metabolomics Data. *Adv. Biosci. Biotechnol.* **9**, 339–351 (2018).

119. Dieterle, F., Ross, A., Schlotterbeck, G. & Senn, H. Probabilistic Quotient Normalization as Robust Method to Account for Dilution of Complex Biological Mixtures. Application in 1H NMR Metabonomics. *Anal. Chem.* **78**, 4281–4290 (2006).
120. Li, B. *et al.* Performance Evaluation and Online Realization of Data-driven Normalization Methods Used in LC/MS based Untargeted Metabolomics Analysis. *Sci. Rep.* **6**, 38881 (2016).
121. Scholz, M., Gatzek, S., Sterling, A., Fiehn, O. & Selbig, J. Metabolite fingerprinting: detecting biological features by independent component analysis. *Bioinformatics* **20**, 2447–2454 (2004).
122. Qannari, E. M., Wakeling, I., Courcoux, P. & MacFie, H. J. H. Defining the underlying sensory dimensions. *Food Qual. Prefer.* **11**, 151–154 (2000).
123. van den Berg, R. A., Hoefsloot, H. C., Westerhuis, J. A., Smilde, A. K. & van der Werf, M. J. Centering, scaling, and transformations: improving the biological information content of metabolomics data. *BMC Genomics* **7**, 142 (2006).
124. Khalheim, O. M. Scaling of analytical data. *Anal. Chim. Acta* **177**, 71–79 (1985).
125. Kasprzak, E. M. & Lewis, K. E. Pareto analysis in multiobjective optimization using the collinearity theorem and scaling method. *Struct. Multidiscip. Optim.* **22**, 208–218 (2001).
126. Keenan, M. R. & Kotula, P. G. Accounting for Poisson noise in the multivariate analysis of ToF-SIMS spectrum images. *Surf. Interface Anal.* **36**, 203–212 (2004).
127. Gromski, P. S. *et al.* A tutorial review: Metabolomics and partial least squares-discriminant analysis – a marriage of convenience or a shotgun wedding. *Anal. Chim. Acta* **879**, 10–23 (2015).
128. Mendez, K. M., Reinke, S. N. & Broadhurst, D. I. A comparative evaluation of the generalised predictive ability of eight machine learning algorithms across ten clinical metabolomics data sets for binary classification. *Metabolomics* **15**, 150 (2019).
129. GOWER, J. C. Some distance properties of latent root and vector methods used in multivariate analysis. *Biometrika* **53**, 325–338 (1966).
130. Xu, Y. *et al.* Application of Dissimilarity Indices, Principal Coordinates Analysis, and Rank Tests to Peak Tables in Metabolomics of the Gas Chromatography/Mass Spectrometry of Human Sweat. *Anal. Chem.* **79**, 5633–5641 (2007).
131. Nguyen, L. H. & Holmes, S. Ten quick tips for effective dimensionality reduction. *PLOS Comput. Biol.* **15**, e1006907 (2019).
132. Jäggi, C., Wirth, T. & Baur, B. Genetic variability in subpopulations of the asp viper (Vipera aspis) in the Swiss Jura mountains: implications for a conservation strategy. *Biol. Conserv.* **94**, 69–77 (2000).
133. Pinheiro, H. P., de Souza Pinheiro, A. & Sen, P. K. Comparison of genomic sequences using the Hamming distance. *J. Stat. Plan. Inference* **130**, 325–339 (2005).
134. Lozupone, C. & Knight, R. UniFrac: a New Phylogenetic Method for Comparing Microbial Communities. *Appl. Environ. Microbiol.* **71**, 8228–8235 (2005).
135. Brejnrod, A. *et al.* Implementations of the chemical structural and compositional similarity metric in R and Python. 546150 Preprint at https://doi.org/10.1101/546150 (2019).
136. Tripathi, A. *et al.* Chemically informed analyses of metabolomics mass spectrometry data with Qemistree. *Nat. Chem. Biol.* **17**, 146–151 (2021).
137. Ramette, A. Multivariate analyses in microbial ecology. *FEMS Microbiol. Ecol.* **62**, 142–160 (2007).

138. Koenig, J. E. *et al.* Succession of microbial consortia in the developing infant gut microbiome. *Proc. Natl. Acad. Sci.* **108**, 4578–4585 (2011).
139. Paliy, O. & Shankar, V. Application of multivariate statistical techniques in microbial ecology. *Mol. Ecol.* **25**, 1032–1057 (2016).
140. Efron, B. Bootstrap Methods: Another Look at the Jackknife. in *Breakthroughs in Statistics: Methodology and Distribution* (eds. Kotz, S. & Johnson, N. L.) 569–593 (Springer, 1992). doi:10.1007/978-1-4612-4380-9_41.
141. Desu, M. M. & Raghavarao, D. *Nonparametric Statistical Methods For Complete and Censored Data*. (CRC Press, 2003).
142. Anderson, M. J. A new method for non-parametric multivariate analysis of variance. *Austral Ecol.* **26**, 32–46 (2001).
143. Anderson, M. J. & Walsh, D. C. I. PERMANOVA, ANOSIM, and the Mantel test in the face of heterogeneous dispersions: What null hypothesis are you testing? *Ecol. Monogr.* **83**, 557–574 (2013).
144. Tibshirani, R., Walther, G. & Hastie, T. Estimating the number of clusters in a data set via the gap statistic. *J. R. Stat. Soc. Ser. B Stat. Methodol.* **63**, 411–423 (2001).
145. Wilkinson, L. & Friendly, M. The History of the Cluster Heat Map. *Am. Stat.* **63**, 179–184 (2009).
146. Wu, W. & Noble, W. S. Genomic data visualization on the Web. *Bioinformatics* **20**, 1804–1805 (2004).
147. Benton, P. H. *et al.* An Interactive Cluster Heat Map to Visualize and Explore Multidimensional Metabolomic Data. *Metabolomics Off. J. Metabolomic Soc.* **11**, 1029–1034 (2015).
148. Ren, S., Hinzman, A. A., Kang, E. L., Szczesniak, R. D. & Lu, L. J. Computational and statistical analysis of metabolomics data. *Metabolomics* **11**, 1492–1513 (2015).
149. Liebal, U. W., Phan, A. N. T., Sudhakar, M., Raman, K. & Blank, L. M. Machine Learning Applications for Mass Spectrometry-Based Metabolomics. *Metabolites* **10**, 243 (2020).
150. Breiman, L. (out-of-bag estimates). (1996).
151. Breiman, L. Random Forests. *Mach. Learn.* **45**, 5–32 (2001).
152. Strobl, C., Boulesteix, A.-L., Kneib, T., Augustin, T. & Zeileis, A. Conditional variable importance for random forests. *BMC Bioinformatics* **9**, 307 (2008).
153. Archer, K. J. & Kimes, R. V. Empirical characterization of random forest variable importance measures. *Comput. Stat. Data Anal.* **52**, 2249–2260 (2008).
154. Jafari, M. & Ansari-Pour, N. Why, When and How to Adjust Your P Values? *Cell J. Yakhteh* **20**, 604–607 (2019).
155. Korthauer, K. *et al.* A practical guide to methods controlling false discoveries in computational biology. *Genome Biol.* **20**, 118 (2019).
156. Mishra, P. *et al.* Descriptive Statistics and Normality Tests for Statistical Data. *Ann. Card. Anaesth.* **22**, 67–72 (2019).
157. Vinaixa, M. *et al.* A Guideline to Univariate Statistical Analysis for LC/MS-Based Untargeted Metabolomics-Derived Data. *Metabolites* **2**, 775–795 (2012).
158. Riffenburgh, R. H. & Gillen, D. L. *Statistics in Medicine*. (Academic Press, 2020).
159. Xia, Y. & Sun, J. Hypothesis Testing and Statistical Analysis of Microbiome. *Genes Dis.* **4**, 138–148 (2017).

160. Sato, T. Type I and Type II Error in Multiple Comparisons. *J. Psychol.* **130**, 293–302 (1996).
161. Bathke, A. The ANOVA F test can still be used in some balanced designs with unequal variances and nonnormal data. *J. Stat. Plan. Inference* **126**, 413–422 (2004).
162. Abdi, H. & Williams, L. Newman-Keuls Test and Tukey Test. *Encycl. Res. Des.* (2010).
163. Ostertagová, E., Ostertag, O. & Kováč, J. Methodology and Application of the Kruskal-Wallis Test. *Appl. Mech. Mater.* **611**, 115–120 (2014).
164. Hecke, T. V. Power study of anova versus Kruskal-Wallis test. *J. Stat. Manag. Syst.* **15**, 241–247 (2012).
165. Dinno, A. Nonparametric Pairwise Multiple Comparisons in Independent Groups using Dunn's Test. *Stata J. Promot. Commun. Stat. Stata* **15**, 292–300 (2015).

7. Data and Code Sharing

The FBMN results are available under the following URL:
https://gnps.ucsd.edu/ProteoSAFe/status.jsp?task=b661d12ba88745639664988329c1363e
Raw and processed that is available through the MassIVE repository (MSV000090156) and Zenodo (https://zenodo.org/records/10051610). All code and software is available through GitHub under the following link https://github.com/Functional-Metabolomics-Lab/FBMN-STATS.

8. Acknowledgment

We thank Libera Lo Presti for critical reading of the manuscript. We thank Greg Caporaso for guidance on preparing the QIIME2 plugins. DP, CM, and HPL were supported by the Deutsche Forschungsgemeinschaft (DFG) through the CMFI Cluster of Excellence (EXC 2124) and DP and CM, were supported by the DFG through the Collaborative Research Center CellMap (TRR 261). KD was supported by the DFG (BO 1910/23). PS was supported by the European Union's Horizon Europe research and innovation programme through a Marie Skłodowska-Curie fellowship No. 101108450 MeStaLeM. TP was supported by the Czech Science Foundation (GA CR) grant 21-11563M and by the European Union's Horizon 2020 research and innovation programme under Marie Skłodowska-Curie grant agreement No. 891397. TD was supported by the MSCA Fellowships CZ (OP JAK) grant CZ.02.01.01/00/22_010/0002733. MW was supported by the National Institutes of Health (NIH) with grants 1U24DK133658-01, NIH 1R03DE032437-01, and UC Riverside startup funding. EEK was supported by grants of the Novo Nordisk Foundation [NNF20CC0035580, NNF16OC0021746]. YW was supported by NIH 1R03DE032437-01. CB was supported by the Czech Academy of Sciences (CAS PPLZ) L200552251. FO was supported by FAPESP 2022/14603-8. JB was supported by Deutsches Zentrum für Infektionsforschung (DZIF). EEK was supported by grants of the Novo Nordisk Foundation (NNF20CC0035580, NNF16OC0021746).

9. Author Contribution

AKPS, FO, FR, ME, and DP conceptualized the protocol. YE, SZ, JS, RS advised on the concept and statistical test. AKPS, AW, FO, FR, MN, JB, EEK, JE, AP, CGM, SF, NC, YW, MD, JS, MW, and ME wrote code. AW, and MW developed and deployed the web app. RS, ATA and DP collected the water samples. DP extracted the samples and acquired the LC-MS/MS data. AKPS, AW, FO, FR, MN, JB, JJK, EEK, JE, AP, CGM, SF, MRA, TP, NC, MP, CB, BC, AMCR, AC, Fd, KD, YE, CG, LGG, MH, SH, SK, AK, MCMK, KM, SP, PWP, TS, KSL, PS, ST, GAV, BCW, SX, MTY, SZ, Md, CB, HPL, CM, JJJvdH, TD, PCD, JS, RS, ATA, ME, and DP tested the protocol, code and app. CB, JJJvdH, TP, MW, ATA, ME, and DP supervised students and researchers. MW, AA, ME, and DP supervised the project. AKPS, MN, JB, JJK, EEK, AP, SF, TP, ATA and DP wrote the manuscript and supplemental information. FO, FR, JE, CGM, MRA, NC, MP, KD, YE, LGG, MH, SH, PS, GAV, SZ, JJJvdH, TD, TP, PCD, JS, RS, MW, and ME edited and provided critical feedback on the first draft. All authors edited and approved the final draft.

10. Conflict of Interest

JJJvdH is currently a member of the Scientific Advisory Board of Naicons Srl., Milano, Italy, and is consulting for Corteva Agriscience, Indianapolis, IN, USA. PCD is a scientific advisor and holds equity to Cybele and a Co-founder, advisor and holds equity in Ometa, Arome and Enveda with prior approval by UC-San Diego and consulted in 2023 for DSM animal health. MW is the founder of Ometa Labs.

11. Additional information & Supplementary information

Supplemental information, including a cheat-sheet, detailed methods for the LC-MS/MS data acquisition and step-to-step guides for the Python and QIIME2 scripts as well as the web app are available in the supplemental information.