nature protocols

https://doi.org/10.1038/s41596-024-01046-3

Protocol

Check for updates

Statistical analysis of feature-based molecular networking results from non-targeted metabolomics data

A list of authors and their affiliations appears at the end of the paper

Abstract

Feature-based molecular networking (FBMN) is a popular analysis approach for liquid chromatography–tandem mass spectrometry-based non-targeted metabolomics data. While processing liquid chromatography–tandem mass spectrometry data through FBMN is fairly streamlined, downstream data handling and statistical interrogation are often a key bottleneck. Especially users new to statistical analysis struggle to effectively handle and analyze complex data matrices. Here we provide a comprehensive guide for the statistical analysis of FBMN results, focusing on the downstream analysis of the FBMN output table. We explain the data structure and principles of data cleanup 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. All code is shared in the form of Jupyter Notebooks (https://github.com/Functional-Metabolomics-Lab/FBMN-STATS). 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 and for educational purposes. Finally, we also show users how to integrate their statistical results into the molecular network using the Cytoscape visualization tool. Throughout the protocol, we use a previously published environmental metabolomics dataset for demonstration purposes. Together, the protocol, code and web application provide a complete guide and toolbox for FBMN data integration, cleanup 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 and can be easily adapted to other mass spectrometry feature detection, annotation and networking tools.

Key points

- Feature-based molecular networking (FBMN) is a popular workflow for liquid chromatography–tandem mass spectrometry-based non-targeted metabolomics data analysis.
- This protocol provides a detailed guide, code (R, Python and QIIME2) and a web application for FBMN data integration, clean-up and advanced statistical analysis, allowing new and experienced users to uncover molecular insights from their non-targeted metabolomics data.

Key references

Nothias, L.-F. et al. *Nat. Methods* **17**, 905–908 (2020): https://doi.org/10.1038/s41592-020-0933-6

Heuckeroth, S. et al. *Nat. Protoc.* (2024): https://doi.org/10.1038/s41596-024-00996-y

Petras, D. et al. *Chemosphere* **271**, 129450 (2021): https://doi.org/10.1016/j.chemosphere.2020.129450

Wegley Kelly, L. et al. *Proc. Natl Acad. Sci. USA* **119**, e2110283119 (2022): https://doi.org/10.1073/pnas.2110283119

Neuhaus, G. F. et al. *PLOS One* **19**, e0303273 (2024): https://doi.org/10.1371/journal.pone.0303273

e-mail: MAET@ssi.dk; dpetras@ucr.edu

Introduction

The field of metabolomics aims to characterize and quantify the detectable spectrum of small molecules 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 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 approach that integrates feature-detection tools with molecular networking (MN) for metabolite annotation and annotation propagation[6] in the Global Natural Products Social Molecular Networking (GNPS) cloud ecosystem[7]. FBMN is routinely applied in many biological disciplines, including clinical studies[8,9], plant[10–12] and environmental science[13–16], as well as microbiome investigations[17–19] and the functional analysis of natural products[20–22]. While platforms such as GNPS have improved the way we identify and characterize metabolites, the statistical analysis of non-targeted metabolomics data remains a challenge for many researchers. Resources such as MetaboAnalyst[23,24] provide powerful solutions for the statistical analysis of metabolomics data but lack shareability and transparency as well as straightforward means of integration with the complex multilayer information from FBMN results and other downstream annotation tools (for example, SIRIUS). Most tools and analysis approaches that can be used to achieve 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 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, promoting an end-to-end pipeline from feature detection, subsequent data clean-up and statistical analysis to spectrum annotation. This step-by-step guide is complemented with ready-made code for the popular statistical scripting platforms R and Python, the Quantitative Insights Into Microbial Ecology 2 (QIIME2) toolkit (https://github.com/Functional-Metabolomics-Lab/FBMN-STATS), a web application with a graphical user interface (GUI) designed to simplify the process (https://fbmn-statsguide.gnps2.org/), and a downloadable GUI application (https://www.functional-metabolomics.com/resources) for new users and educational purposes. The protocol can be seamlessly integrated with FBMN results from GNPS (https://gnps.ucsd.edu) and GNPS2 (https://gnps2.org).

FBMN from LC–MS/MS data

Liquid chromatography (LC)–mass spectrometry (MS) is one of the most prominent metabolomics techniques, with applications in numerous research fields[25–28]. Specifically, LC–MS/MS has been widely used because it provides a broad coverage of chemical space allowing for the simultaneous semiquantitative detection and qualitative annotation of many metabolites, including both organic and inorganic compounds, over a wide dynamic range[14,29–32].

In addition to providing the molecular mass, retention time (RT) 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

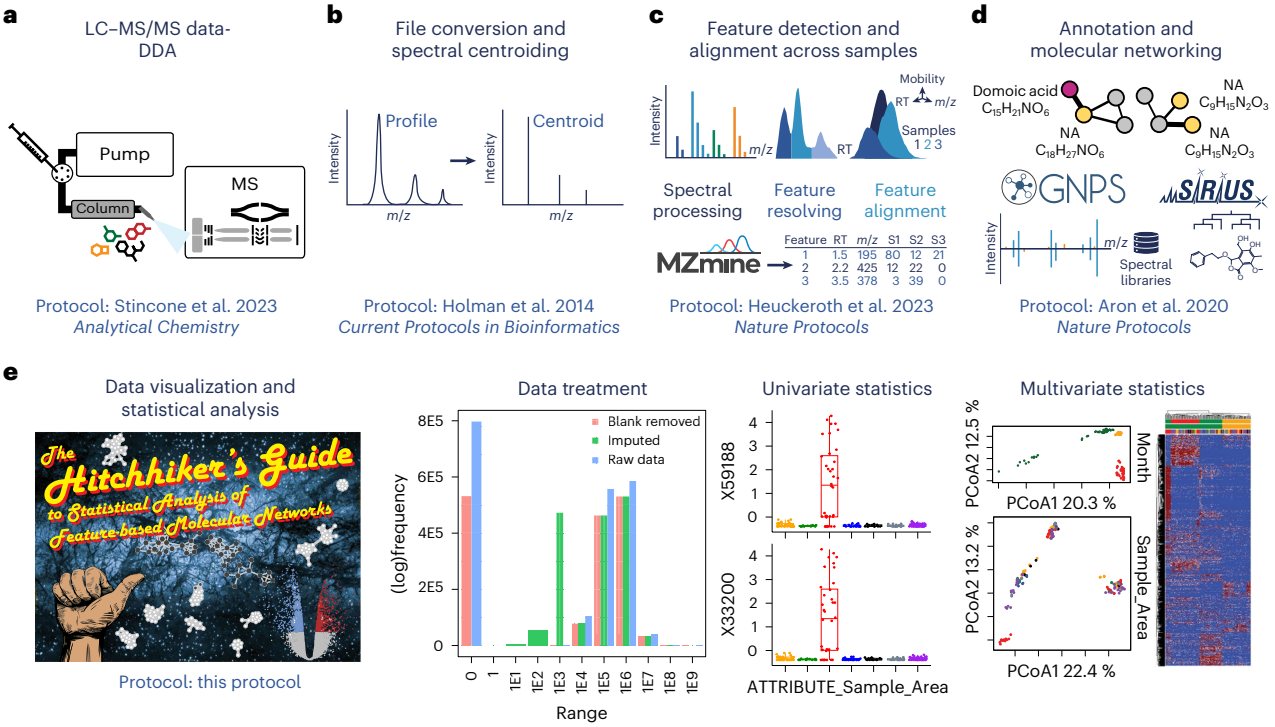


Fig. 1 | Flowchart of LC–MS/MS-based metabolomics experiment. a, DDA of MS/MS spectra. **b**, Centroiding and file conversion. **c**, Feature detection. **d**, Feature annotation, network propagation and clustering. **e**, Data clean-up, statistical analysis and visualization (blank removal, imputation, normalization and scaling, followed by data visualization and statistical analysis).

mass-to-charge ratios (*m/z*) and abundances. This can be conducted using data-dependent acquisition (DDA) or data-independent acquisition (DIA) methods. DDA involves selecting ions observed in full-scan MS (MS1) for further fragmentation in subsequent MS/MS scans (Fig. 1). It operates by selecting the top *N* peaks in each duty cycle, where *N* is a user-defined number. These peaks are chosen on the basis of their intensity in a narrow isolation window and other user-defined criteria through an automated process[15]. Conversely, the DIA method fragments all ions within a predefined larger and sequential isolation window during each scan, directing the fragmented ions to a high-resolution MS analyzer. Consequently, the MS/MS spectra show high complexity that demands the use of advanced processing and annotation tools, that have only recently become available to the scientific community. So far, most of the non-targeted metabolomics studies are performed in DDA mode due to its generation of simpler data and more straightforward analysis procedures compared with DIA[15].

The resulting MS/MS spectra of product ions from either method can be used in several ways to indicate 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 the 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 of non-targeted metabolomics. Qualitative data exploration has been democratized by platforms such as GNPS[38], by providing MS reference libraries,

BOX 1

Experimental design for LC–MS/MS data acquisition

To obtain high-quality and representative LC–MS/MS data, proper planning of the sampling and MS analysis is essential. While this protocol article focuses on data analysis, the integrity and reliability of the results are heavily influenced by meticulous sample handling to avoid introducing bias. No single platform is capable of encompassing the entire metabolome. Therefore, the selection of an analytical approach should align with the specific research objectives and the sample characteristics[143]. We strongly emphasize the importance of rigorous sample preparation for achieving reliable and robust outcomes from the FBMN. For researchers new to these types of experiments, we advise seeking guidance from a statistician and analytical chemists to guarantee optimal experimental design, instrument performance (for example, system suitability tests), and analysis pipeline. Before proceeding with further processing and statistical analysis, raw LC–MS/MS data should be manually inspected by the user[108,144,145]. 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, such as 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[146] solvent and extraction blanks. A solvent blank consists of only the solvent used to dissolve the sample and is used to identify the contaminants present in the solvent. 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. An extraction blank 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 with the real sample to find the contaminants introduced during the extraction process.
- Control samples, for example, 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.
- 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, for example, knockout versus wild type mutant strains or low vs. high biomass samples, to avoid carryover between them.
- Managing batch effects: 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[114]. 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 postsample measurement[147]. 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[148]. See the section on batch correction for more details.
- Internal standards 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.

data analysis workflows and compute resources for the community. 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 MN by connecting ions from the same molecules into groups called ion identity networks. This helps remove redundancy in MS-based metabolomics.

FBMN workflow

This protocol addresses the downstream analysis of FBMN results from non-targeted metabolomics data. This section offers a structured approach leading up to FBMN and before our statistical workflow. As highlighted in Fig. 1, the non-targeted LC–MS/MS analysis workflows can be split into five main stages:

1. Data acquisition
2. File conversion with centroiding of the raw data
3. Feature detection and an optional ion identity networking
4. Feature annotation, which encompasses spectral library matching, in silico spectrum annotation, annotation propagation and spectral clustering methods such as spectral networking
5. Data refinement, visualization and statistical analysis

Stages 1 to 4 result in a comprehensive feature table, capturing all detected features (m/z, RT, and MS/MS spectra) alongside quantitative information (for example, peak area) for each sample, including the feature annotations. The focus and main work of this article is predominantly on the fifth stage, involving the refinement of the feature-sample matrix through clean-up steps such as blank removal, imputation, normalization, scaling and ultimately, data visualization and statistical analysis. Accordingly, the emphasis is on guiding users through stage 5, but it is crucial to be aware of how the preceding processing can influence results. We encourage users to review the protocols outlined in Fig. 1 for a comprehensive understanding of the processing stages taken to produce the feature table used in our protocol. We also briefly discuss each stage, as this information serves as a foundational starting point for the remainder of the protocol.

1. Data acquisition

Data acquisition very much depends on the experiment. Some guidelines on good experimental design can be found in Box 1.

2. File conversion

Raw data acquisition in MS instruments involves first generating spectra in profile mode, also called continuous mode, typically resulting in a Gaussian shape. In high-resolution instruments, each chemical entity is represented by signals of m/z values within a 5–20 ppm window, depending on the instrument's resolution. Researchers with access to vendor-specific software are encouraged to examine the raw data directly from these platforms to assess data quality. To simplify data for downstream analysis, a process called centroiding or 'peak-picking', is employed; this process converts each Gaussian peak in the m/z dimension into a singular peak, thereby reducing data complexity. Centroiding can be performed on-the-fly during data acquisition, where the profile information is lost or postacquisition through file conversion tools such as the Proteowizard's msconvert[40] or the Thermo dedicated ThermoRawFileParser[41]. For novices, msconvert offers an accessible starting point, supporting binary files from various vendors with an intuitive GUI. When selecting the 'vendor' option for centroiding in msconvert, the algorithm provided by the vendor of the instrument is applied[42]. Nonetheless, vendor-specific tools, such as ThermoRawFileParser, may offer more precise algorithms than msconvert's default, making them preferable for accuracy (Fig. 1) as they can include the removal of flagged, for example, noise or internal calibrant, signals[41]. Note that the conversion of vendor-specific formats into universally accessible, open formats such as 'mzML' is often needed for compatibility with most feature detection software and to make data accessible without vendor-specific tools.

3. Feature detection

The process of converting raw data from the preceding stage into a table of putative metabolic features, along with their relative abundances per sample, involves a preprocessing workflow that uses a series of algorithms (Fig. 1). 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[43] (often used with the package CAMERA[44] for feature grouping), MZmine 3[45], MS-DIAL[46] or OpenMS[47], in addition to vendor-specific tools can be utilized for this purpose. For advanced optimization and fine-tuning, multiple tools such as NeatMS[48], MetaClean[49] and mzRAPP[50] exist to assess feature quality.

For the present protocol, we chose MZmine 3 for feature detection for the following reasons. First, it provides an interactive and user-friendly GUI that can assist researchers without programming skills. Second, harmonized data exchange formats enable its tight integration with downstream annotation tools such as GNPS and SIRIUS. Third, MZmine offers a tool (called 'Processing wizard') for the automatic generation of data-processing workflows that can be used by new users as a starting point for parameter optimization[51]. For detailed descriptions and recommendations for processing parameter optimization, we direct the reader to the recently published MZmine 3 protocol[51] and the software's online documentation (https://mzmine.github.io/mzmine_documentation/index.html, https://mzmine.github.io/). Here, we want to emphasize that suboptimal parameters can dramatically impact the quality of the resulting feature quantification table and impair all downstream steps. For instance, a feature table with more than 20,000 features might notably prolong or even cause the FBMN process to fail. Therefore, during the feature detection phase, distinguishing features from noise is crucial, such as ensuring the features fall above the noise threshold. This necessitates particular attention, especially from new users at this stage.

Although the present protocol uses MZmine 3 as feature detection software, users of alternative platforms such as MS-DIAL, XCMS and OpenMS can also integrate their data using their FBMN task identification. FBMN accommodates input tables from these platforms, standardizing them for the workflow. Utilizing an FBMN identification ensures the feature table automatically conforms to our protocol's requirements. For those inputting data manually, we advise referencing our example dataset to understand necessary adjustments, thereby allowing for tailored adaptation of their data to meet protocol requirements.

4. Feature annotation, spectral networking and annotation propagation

The metabolomics standard initiative outlines four levels of metabolite annotation and structural identification through MS to guide researchers in differentiating the level of identification rigor for the reported metabolites[52]. Level 1 annotations require fully characterized compounds such as authentic standards, analyzed under identical conditions as the experimental samples. To ensure accurate annotation confidence at level 1 for a feature, it is necessary to match two or more independent orthogonal data dimensions to those of an analytical standard. These dimensions typically include precursor m/z (MS1), RT and MS/MS fragmentation patterns and notable differences in any of the available data dimensions preclude level 1 annotation. Generally, a feature is only described as 'identified' when a level 1 annotation was achieved while for those with lower confidence, the term 'annotation' is preferred.

In the context of MS/MS, level 2 confidence is assigned when data are confidently matched to reference MS/MS spectra (precursor and fragment m/z values)[52], as facilitated through, for example, FBMN on GNPS. The assignment of level 2 annotations usually requires manual curation of spectral matches, since spectral matching scores are not fool-proof and especially scores between spectra of different compounds from classes with high isomerism and structural analogism, such as steroids and other terpenoids, can produce highly similar MS/MS spectra. Moreover, spectral libraries contain entries with insufficient fragmentation to allow level 2 annotation, such as matching on the basis of low-intensity fragments and the residual precursor ion. Matching to such entries can be avoided by tuning parametrization of the FBMN workflow, such as the minimum number of matched fragments and noise level filters, but more stringent requirements may lead to losses of acceptable matches.

Table 1 | Example of a feature quantification table as generated by MZmine

	m/z	RT (min)	Adduct	Charge	Sample 1	Sample 2	...	Sample *N*
Feature 1	97.1082	4.6	$[M+H]^+$	+1	2.08×10^{-7}	9.47×10^6	...	3.27×10^8
Feature 2	518.3032	2.0	$[M+H]^{+2}$	+2	1.88×10^7	5.56×10^5	...	2.11×10^6
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Feature K	83.1017	1.6	$[M+Na]^+$	+1	4.77×10^4	8.13×10^3	...	5.17×10^9

The table illustrates how a typical quantification output from MZmine appears. Adducts and charges are assigned by running specific modules (de-isotoper and ion identity networking modules) in MZmine[39,45].

Level 3 annotations are given to features for which chemical classes can be inferred through molecular network connectivity with other annotated features, analog spectral matching to reference MS/MS spectra (rational *m/z* deviation and high similarity score) or in silico prediction tools such as SIRIUS[36,53], CSI:FingerID[35] or CANOPUS[37,54]. The results of these approaches can be contradictory for the same feature and manual curation is required to maintain good confidence. Finally, level 4 refers to unknown features that can be consistently detected (for example, defined *m/z* value, RT and MS/MS spectrum) but not annotated to any of the higher levels.

Feature annotation is essential in MS-based metabolomics studies, especially to understand the biological importance 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, Dereplicator and Dereplicator+, predict metabolite identities based on spectral similarities[53] and can only lead to metabolomics standard initiative levels 2, 3 or 4.

Another popular method that combines feature annotation through spectral similarity scoring with visualization is MN, as shown in Fig. 1. MN elucidates the structural relationships between metabolites based on similarities in their fragmentation patterns, highlighting potential biological pathways and processes. The utility of MN spans across various fields, such as natural products[55], agriculture[56] and clinical microbiology[57]. Using the MS-cluster algorithm on the GNPS (https://gnps.ucsd.edu) and GNPS2 (https://gnps2.org) web platforms, MN generates a molecular network by calculating spectral similarities between each MS/MS spectra pair and provides structural annotations of varying reliability[7] by scoring spectral similarity against public spectral database entries. The .mzML or .mzXML spectra files can be analyzed directly on GNPS using the classical MN workflow[38], which is based solely on the comparison of MS2 spectra in a dataset without any inflow of chromatographic information from the MS1 dimension.

For more precise quantitative insights, FBMN has emerged as a major advancement by incorporating MS1 peak intensities, retention times, isotope patterns and, when applicable, ion mobility separation. Consequently, FBMN distinguishes between isomers with near-identical fragmentation spectra but different retention times[6] and allows feature peak areas to feed into the data exploration. FBMN on the GNPS web platform conveniently accepts the output of feature detection and alignment tools such as Mzmine[45,58] (see 'Feature detection' section), MS-DIAL[46] and XCMS[43] that assemble feature lists and associate feature fragmentation spectra from raw MS/MS data.

5. Data refinement, visualization and statistical analysis

The feature quantification table ('Feature detection' section and Table 1), contains a list of features, such as *m/z* and RT values, as well as their relative abundances per sample. This table represents the basic 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. In part, chemical and biological relevance can be understood through the FBMN workflow and our protocol offers a toolset to augment an FBMN analysis with information on the statistical significance of feature distribution and sample compositions.

To refine the feature quantification table, the protocol involves clean-up steps such as blank removal, imputation, normalization and scaling. After data refinement, we apply multivariate

and univariate statistical analyses for a deeper exploration of samples. The multivariate techniques showcased in our workflow include principal coordinates analysis (PCoA), hierarchical clustering analysis (HCA), heat maps and random forest (RF). The univariate techniques involve parametric tests, such as analysis of variance (ANOVA) and *t*-tests, as well as nonparametric counterparts, such as the Kruskal–Wallis (KW) test.

PCoA

Initially, dimensionality reduction techniques, such as principal component analysis (PCA) and PCoA, are employed for data exploration and visualization. These unsupervised techniques generate two- or three-dimensional plots grouping similar samples, despite starting with a high number of dimensions (up to thousands of features). The key insight is that only a few top dimensions are needed to visualize the most critical data aspects, effectively reducing complexity. This can give valuable impressions on the presence of sample clusters and how they relate to metadata categories[59].

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 focuses on their correlation or covariance[60]. 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[61]. It is important to mention that MDS can be categorized into metric MDS (as in PCoA) and nonmetric MDS[60]. In this protocol, we focus solely on metric MDS.

Coupled with permutational multivariate ANOVA (PERMANOVA), these techniques enable a comprehensive exploration of sample similarity by calculating correlations or distance matrices. Although not demonstrated in our workflow, *t*-distributed stochastic neighbor embedding is another widely used technique for dimension reduction, recognized for its effectiveness in managing complex datasets[62]. For advanced feature extraction in high-dimensional data, tools such as knowledge discovery by accuracy maximization (KODAMA) can be considered. It functions as an unsupervised learning algorithm to perform feature extraction from noisy and high-dimensional data[63].

PERMANOVA

In multivariate analysis such as PCA, it is crucial to measure confidence in observed relationships or separation between samples. 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[64]. In such cases, resampling methods such as bootstrap, jackknife[65] and permutation tests[66] 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[67].

Alternatively, nonparametric methods such as PERMANOVA can be used[68]. PERMANOVA allows for multivariate ANOVA and tests for differences between sample groups. It enables any dissimilarity metric and calculates a test statistic by comparing the dissimilarities between samples within and between groups. Here, the *P* values are determined through permutation[64].

HCA and heat maps

Another unsupervised approach commonly used in metabolomics is the use of hierarchical clustering to group samples with similar relative abundance profiles of features. 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 in combination with a heat map[61]. This combination is ideal for hypothesis generation by providing an initial data overview. The heat map displays the feature distribution across samples in a two-dimensional plot. Here, cells are colored according to the relative abundances of features in samples, with features and/or samples arranged in rows and columns grouped according to

their similarity profiles. A dendrogram is drawn beside the heat map to further illustrate the hierarchical relationship between the samples and features.

Compound class ontologies such as ClassyFire[69] or Natural Product Classifier[70] categorize compounds on the basis of shared structural features or biosynthetic origins and serve as high-level annotations of the data. CANOPUS[54] predicts these compound classes from MS/MS data without searching in structure databases. Analyzing the distribution and variety of compound classes, along with their up- and downregulation using these data visualization methods, can yield biological insights that may not be attainable when solely considering *m*/*z* and retention time values.

However, caution is advised with unsupervised methods; it is essential to critically assess the clusters and dimensions produced. For instance, heat maps can help validate the clusters of HCA by revealing inconsistencies or alignments among samples and features. While users typically select the number of clusters or dimensions for analysis, heuristic methods exist to suggest optimal number of clusters, though they are not definitive[71].

In addition to that, traditional cluster heat maps also have limitations. Their data representation in a 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, heat maps 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 heat maps are available that offer a more versatile and dynamic data visualization experience[72]. For further details on applying these unsupervised and supervised methods, including their advantages and limitations, refer to the respective sections under the 'Procedure' section.

Supervised classification: RF

We use RF as a key supervised classification technique in this protocol. While previously discussed unsupervised methods allows for the discovery of groups or trends in the data without prior assumptions about predetermined labels or categories, supervised analysis uses 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)[73].

Supervised learning is categorized into classification and regression problems based on the type of response variable: classification for categorical or discrete variables (for example, cancer versus noncancer samples) and regression for continuous variables. Popular supervised models in metabolomics include logistic regression, partial least square discriminant analysis (PLS-DA), support vector machines, *k*-nearest neighbor (KNN) and RF. Here, we focus on RF due to its advantages such as the low risk of overfitting, ease of implementation, interpretability and minimal hyperparameter tuning requirements[74].

However, to maintain simplicity, we are not discussing the hyperparameter tuning in the main protocol. For advanced users interested in exploring beyond RF, we offer instructions on using the gradient boosting method via XGBoost, which includes tuning various hyperparameters. XGBoost employs gradient boosting to sequentially correct errors in a series of decision trees, offering a different approach to model enhancement. These additional instructions are provided in a separate Jupyter Notebook (https://colab.research.google.com/github/Functional-Metabolomics-Lab/FBMN-STATS/blob/main/R/Additional_Notebooks/XGBoost_Tuning_Parameters.ipynb) in our GitHub repository. Additionally, we would like to point to 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 such as 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[75]. While we do not dismiss the utility of PLS-DA, we suggest considering alternative models as well. For a comprehensive comparison of different machine learning-based classification tools, we recommend the study by Mendez et al.[76], in which they evaluate eight machine learning algorithms across ten clinical metabolomics datasets for binary classification[76].

Univariate statistics

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 is 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 can use ANOVA. 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.

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[77]:

- 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: an iteration of BH that is suitable when tests have dependencies
- Storey's q value: this approach estimates the proportion of true null hypotheses (that is, no effect) among all hypotheses and then computes a q value for each test, which is the FDR analog to the P value[78]

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 potentially biologically relevant. However, it is important to note that differences observed at this stage are statistical rather than definitive biological distinctions. Follow-up experiments are necessary to confirm these as genuine biological differences. In all our univariate tests, we apply the BH metric to our P values, aiming to robustly identify statistical differences in the data.

Testing for normality is often one of the first steps in univariate analysis and is crucial in deciding whether to use parametric or nonparametric tests. Parametric tests, such as *t*-test or ANOVA, assume that the data follow 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. Shapiro–Wilk is more suitable for small sample sizes ($N < 50$)[79]. Here, 'normal' applies to the entire population and not just the sample data. In these normality tests, the null hypothesis states that the data follow a normal distribution. The resulting P value from such tests is compared with a prespecified α level (for example, 0.05). Here, $P < 0.05$ rejects the null hypothesis, suggesting that the data do not follow a normal distribution. However, $P \geq 0.05$ does not provide any evidence against the null hypothesis given the available data, so there is no evidence that the data are not normally distributed.

Normality becomes less critical with large samples because of the central limit theorem. In such cases, one can consider doing parametric tests instead. When the data do not follow a normal distribution, one can follow nonparametric tests, such as the Mann–Whitney *U* test or the KW test[79].

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 (Fig. 1) that are straightforward, transparent and flexible. We aim to complement the structural insights gained from FBMN with downstream statistical findings, thereby enhancing the understanding of the molecular network. Integrating FBMN results with statistical analyses poses several challenges, often necessitating users to reformat, upload and process the feature table with different external tools to ultimately manually combine the outcomes. Our approach addresses this gap by offering a detailed guide and comprehensive solution to directly process and analyze the data after FBMN in one pipeline, as shown in Fig. 2.

This protocol provides a transparent, user-friendly and versatile approach. Transparency and reproducibility are ensured through shareable notebooks and compiled results. To promote usability, the pipeline is provided in popular scripting languages, R and Python, and is presented through Jupyter Notebooks for local use and Google Colab notebooks for cloud-based applications. The pipeline is highly adaptable as well, allowing users to modify and apply the code according to their specific needs. This flexibility is demonstrated by its recent application in a publication on environmental metabolomics, which utilized codes from our Jupyter Notebooks[80]. The protocol is also beginner friendly, helping new users to easily learn and use Jupyter Notebook for statistical analyses, thus lowering barriers for those not familiar with R or Python. Finally, tool versatility is provided through customizable workflow modules that can be used as part of the pipeline or in combination with other feature detection, annotation and data analysis tools, with or without the use of FBMN.

In addition to the R and Python notebook, this pipeline is also provided within the widely used QIIME2 framework. QIIME2 is a widely recognized next-generation microbiome bioinformatics platform, designed to enable sophisticated analyses in microbiome science[81]. It offers a plugin-based architecture supporting a wide range of functionalities, from sequence analysis to machine learning. QIIME 2 enables comprehensive data integration across various types, such as metabolomics and metagenomics, supported by robust visualization tools for in-depth data exploration[81]. Acknowledging its widespread use, our protocol provides a streamlined workflow within a Jupyter Notebook, specifically designed to facilitate analysis for users already familiar with QIIME 2. More information on this can be found in Supplementary Information.

Additionally, we have developed a web application with a GUI, which can be accessed at https://fbmn-statsguide.gnps2.org/ or downloaded as standalone GUI application from https://www.functional-metabolomics.com/resources. The main manuscript focuses on the concepts and step-to-step guide for the R workflow, while Supplementary Information contains step-to-step guides for the Python, QIIME2 and Web app workflows. Though most steps are consistent across the workflows, any differences are addressed and complemented with alternative solutions in Supplementary Information. 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 and 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 and enables straightforward sharing of statistical data analysis workflows and results

As inputs, the protocol requires a feature table and its corresponding metadata table. For specifications on the layout and requirements of these tables, please refer to Fig. 2 and the

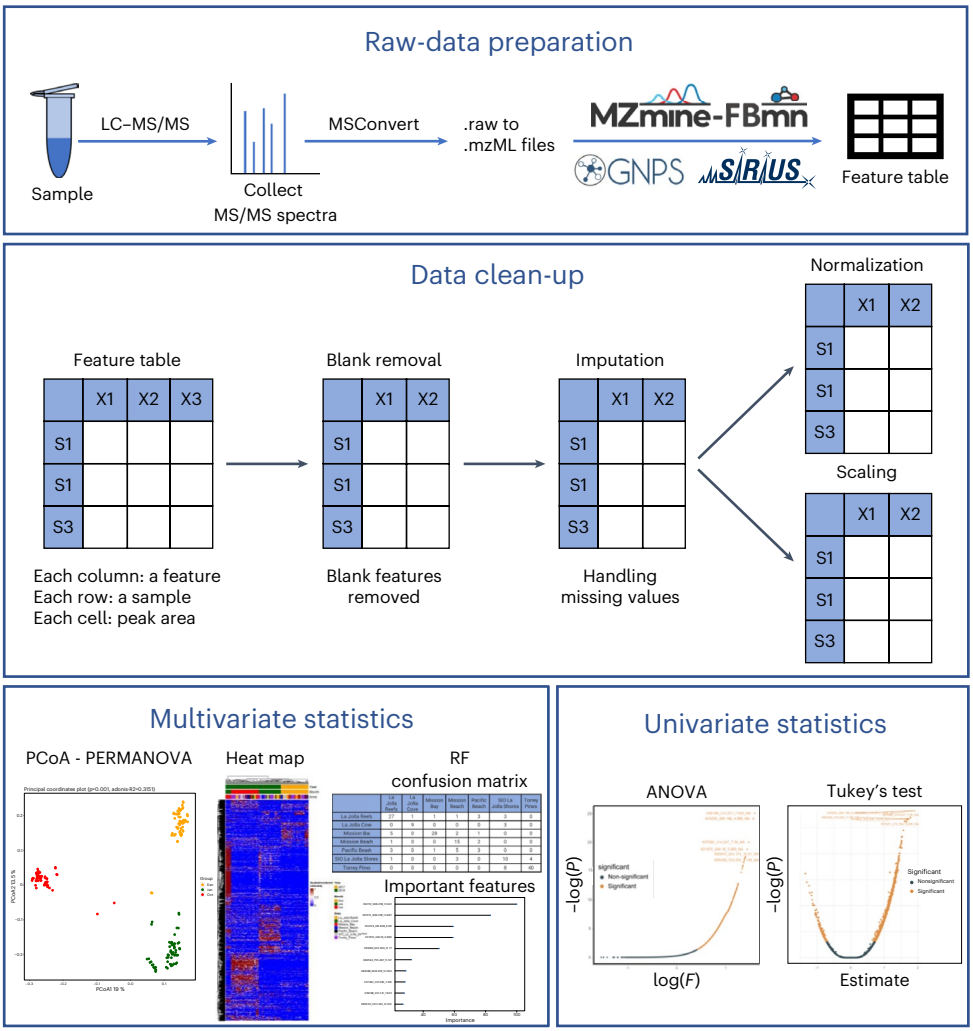


Fig. 2 | Overview of the data analysis pipeline. Integrating four core segments, the flowchart starts with sample collection and LC–MS/MS data acquisition, transitions to raw data conversion into mzML format and results in generating a feature quantification table under 'Raw data preparation'. Note 'Raw data preparation' is not part of the current protocol. More information on this can be found in the first three blocks of Fig. 1. The current protocol commences with the 'Data clean-up' phase, which requires a feature quantification table and an associated metadata table produced in the 'Raw data preparation' section. The feature table contains the relative intensity values for all detected features across the measured samples. The supplementary metadata table contains information on the different categories of the measured samples. The 'Data clean-up' phase emphasizes refining the features on the feature table through blank removal, imputation and normalization strategies such as TIC, PQN and scaling. Subsequently, the "Multivariate Statistics" segment showcases techniques such as Principal Coordinate Analysis (PCoA) plots and heat maps for effective data portrayal. In addition, the users are introduced to robust techniques including RF classification. In the 'Univariate statistics' section in Introduction, tests such as ANOVA and Kruskal–Wallis test are discussed.

'Required files' section in 'Materials'. The feature quantification table, a product of feature detection software, contains a matrix of the relative intensities measured for each feature within each sample. The metadata table provides essential information about the samples and the conditions under which they were measured. Throughout its execution, users receive:

- Intermediate tables after each data clean-up step, aiding in comparison with the original feature table. Tabular outputs for clustering results from HCA, a list of statistically significant features as determined by ANOVA or KW tests and RF 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, heat maps, volcano plots for significance tests and box plots, showcasing group differences for significant features

This protocol also helps with mapping the results of some of the statistical approaches (for example, clustering and 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 to examine the molecular network ('Integrating statistical results into a molecular network' section in Procedure). Moreover, as all our resources are publicly available at https://github.com/Functional-Metabolomics-Lab/FBMN-STATS, users can actively raise issues or provide suggestions there.

Limitations and challenges

Our protocol for FBMN is aimed at offering advanced statistical analysis solutions for a broad range of users. We, thus, offer notebooks and code in different scripting languages (R, Python and QIIME2) and platforms (Jupyter and Colab), as well as a web application to suit the specific needs and preferences within the metabolomic community. Readers are encouraged to refer to Figs. 3 and 4 for a quick overview of the available options and guidance on selecting the most suitable one based on their familiarity with the tools and programming languages and the size of their datasets.

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. Despite these challenges, we primarily focus on R in the main protocol based on R's extensive range of statistical packages and its widespread use in statistical analysis.

On the other hand, installing packages in the Python Google Colab notebook is relatively faster. While Python excels in machine learning applications, it has fewer statistical packages than R. One vital point to note is the incompatibility of the python package 'scikit-bio' with Windows. Thus, Windows OS users are advised to either use the Google Colab version or consider the Windows Subsystem for Linux 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 min 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.

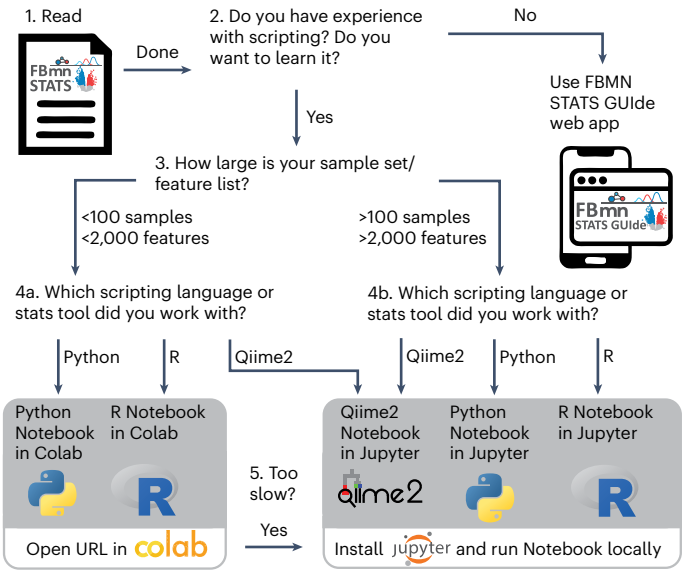
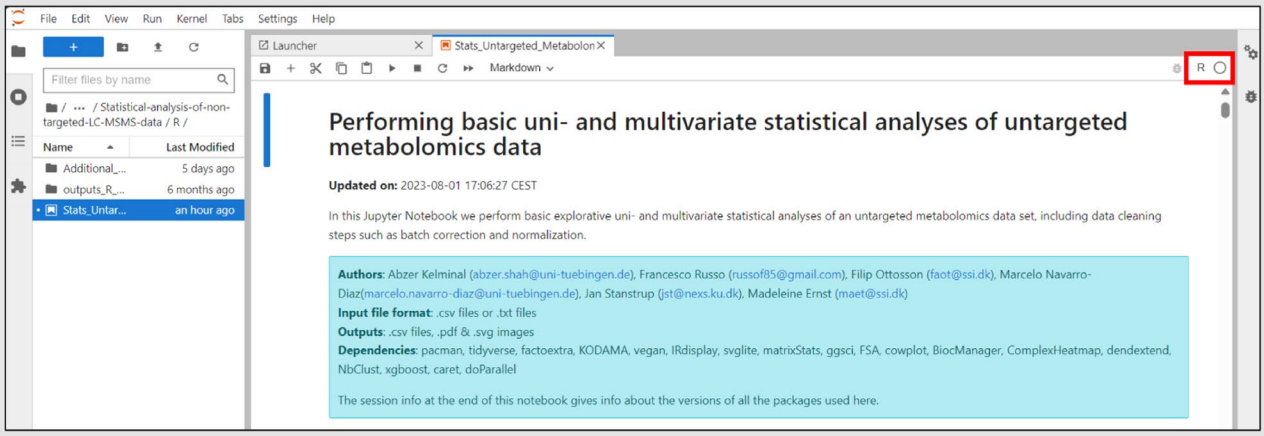


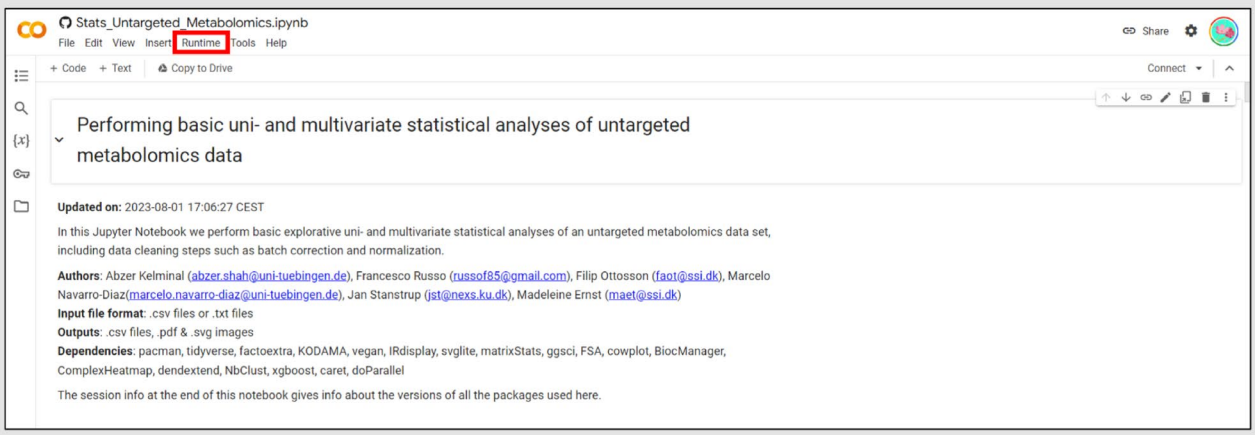
Fig. 3 | Decision tree to guide choosing which notebook or app to use. The number of samples and features mentioned here is for conceptual guidance only and should not be considered as fixed thresholds.

a Screenshot of Jupyter Notebook within JupyterLab interface

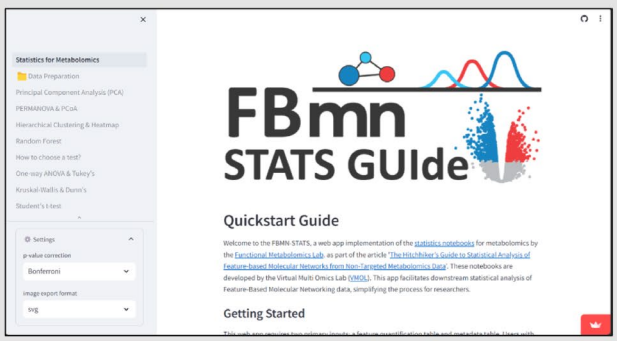


Jupyter Notebook installation guides:
https://github.com/Functional-Metabolomics-Lab/FBMN-STATS/Jupyter-Notebook-Installation-Guides

b Screenshot of Jupyter Notebook in a Google Colab environment



c Screenshot of the web interface



URL: https://fbmn-statsguide.gnps2.org

d Guide to documentation for available tools

Documentation of tools	Location
R notebook	Main protocol
Python notebook	SI
QIIME2 notebook	SI
Web app	SI
Google Colab	GitHub, main protocol

SI, Supplementary Information

Fig. 4 | Interface previews and documentation guide. a, Screenshot of JupyterLab Interface with a Jupyter Notebook, running locally, highlighting the kernel selection area (R, Python) in red. Guide for Jupyter Notebook installation is available on our GitHub repository (https://github.com/Functional-Metabolomics-Lab/FBMN-STATS/tree/main/Jupyter-Notebook-Installation-Guides). **b**, Screenshot of Google Colab Interface with a Jupyter Notebook. Here, the 'Runtime' tab for kernel management is emphasized in red. Colab facilitates cloud-based notebook execution without local installations. **c**, Screenshot of the web application interface of FBMN-STATS (https://fbmn-statsguide.gnps2.org/) along with the URL entry point. **d**, Documentation locations table, providing a centralized reference for where to find guidance for each tool, which is dispersed across the main protocol, Supplementary Information and README files in the FBMN-STATS GitHub repository (https://github.com/Functional-Metabolomics-Lab/FBMN-STATS).

Both the R and Python notebooks comprise over 80 steps, with a substantial 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 (for example, those with over 100 samples and more than 2,000 features), especially given the constraints of Google Colab. Despite this, the protocol still functions for large datasets, albeit with increased complexity and computational demands.

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 or follow the additional installation guide provided in the GitHub repository (https://github.com/Functional-Metabolomics-Lab/FBMN-STATS/tree/main/Jupyter-Notebook-Installation-Guides). The Streamlit web application for the protocol, although user friendly, has its own set of challenges. Notably, there is a data restriction of 200 MB, and larger datasets might inadvertently slow down the app or even lead to server crashes. Additionally, it offers less flexibility and fewer graphical data representation options compared with comprehensive tools such as MetaboAnalyst[23,24]. Although it serves as an introductory tool to the concepts, the notebooks facilitate more tailored analyses for specific research questions. For complex and detailed investigations, scripting is indispensable. Thus, the GUI is optimally employed as an educational tool, with more advanced resources recommended for in-depth research applications.

Lastly, the QIIME2 notebook is broadly used and applicable for both the microbiome and metabolomics communities. This 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[81]. 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.

Although we highlight the importance of experimental design and data acquisition for reducing biological and analytical biases (Box 1), we acknowledge certain limitations in our dataset and workflow, such as the absence of guidance on power calculations. We also acknowledge that our protocol does not include uncertainty propagation, a process important for detailed statistical analysis that tracks how initial uncertainties, such as measurement errors, influence the final results. This omission is mainly due to the immense computational resources needed for such estimation in high-dimensional metabolomics data. Despite discussing the limitations of univariate and multivariate analysis methods, our protocol prioritizes data exploration and hypothesis generation, where immediate precision in uncertainty estimation is not critical[82].

Alternative open-source data analysis workflows and protocols

There have been many efforts in the community to provide and teach statistics solutions 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 point out the many other tools, workflows and applications that are available.

Table 2 summarizes what are, in our opinion, the most commonly used open-source software tools. The table provides an overview of their functions, purpose, tool type and, when applicable, references to related protocols and guidelines. Additionally, the table includes a column named 'Estimated runtime for respective sample data', offering insights into the processing time required by each tool for the sample data they provided or an estimation for our sample data. We present these times in terms of minutes, hours and days, also noting potential bottlenecks. This information aims to give users a rough benchmark while acknowledging the diversity in sample data and computational resources across different tools. Further details can be explored through the references listed in the 'Reference' column. 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,

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

Tool Name	Tool Type	Raw data processing	Blank removal	Matrix transformations	Univariate statistics	Multivariate statistics	Export for downstream tools	Customizable	Estimated runtime for respective sample data	URL	Reference
GUI											
MetaboAnalyst	Web App (GUI)	Y	Y	Y	Y	Y	Y	N	Hours (bottleneck: raw data processing)	www.metaboanalyst.ca/	23,24
Workflows											
Galaxy-M	Workflow	Y	Y	Y	Y	Y	N	N	Hours (bottleneck: raw data processing)	github.com/Viant-Metabolomics/Galaxy-M	43,131
Workflow4-Metabolomics	Workflow	Y	Y	Y	Y	Y	N	N	Hours (bottleneck: raw data processing)	github.com/workflow4metabolomics	132
UmetaFlow	Workflow	Y	Y	Y	Y	Y	N	N	Days (bottleneck: formula and structural predictions)	github.com/biosustain/snakemake_UmetaFlow	133
Chemometrics tutorials	Workflow/tutorial	N	N	Y	Y	Y	N	Y	Hours (bottleneck: power analysis)	github.com/Gscorreia89/chemometrics-tutorials	
QIIME2 metabolomics plugin	Language	N	N	N	Y	Y	N	N	NA	library.QIIME2.org/plugins/q2-metabolomics/10/	81
R libraries											
mixOmics	Package	N	N	Y	Y	Y	Y	Y	Minutes (bottleneck: parameter tuning)	mixomics.org/	134
MetaboanalystR	Package	Y	Y	Y	Y	Y	Y	Y	Minutes–hours (bottleneck: raw data processing)	www.metaboanalyst.ca/docs/RTutorial.xhtml	135,136
omu	Package	N	N	Y	Y	Y	Y	Y	NA	cran.r-project.org/web/packages/omu/vignettes/Omu_vignette.html	137
metabolomicsR	Package	N	N	Y	Y	Y	Y	Y	NA	github.com/XikunHan/metabolomicsR	138
MAIT	Package	Y	N	Y	Y	Y	Y	Y	Hours (bottleneck: raw data processing)	bioconductor.org/packages/release/bioc/html/MAIT.html	139
ropls	Package	N	N	Y	Y	Y	Y	Y	NA	bioconductor.org/packages/release/bioc/html/ropls.html	140
MSStats	Package	N	Y	Y	Y	Y	Y	Y	Minutes (bottleneck: data cleaning)	github.com/Vitek-Lab/MSstats	141
Python libraries											
TidyMS	Package	Y	Y	Y	N	N	Y	Y	Minutes (bottleneck: data curation pipeline)	github.com/griquelme/tidyms	142

All tools listed here are open source and freely available.

see 'Feature detection' section), data clean-up steps (involving quality control, missing value imputation, normalization, scaling and transformation) and multivariate and univariate analyses.

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[83] and typically require specific file and table formats.

Acknowledging the extensive array of existing platforms, the unique value of our protocol lies in its versatility and accordance with FAIR research principles[84]. Processed feature tables generated after each step, such as post-blank removal, normalization and scaling, can be integrated with other analysis software, such MetaboAnalyst, or amended with newly created scripts. Additionally, the Jupyter Notebooks provide easily shareable and reproducible data analysis records, thereby promoting maximum transparency. Our protocol notably reduces the learning curve associated with Jupyter Notebook, making advanced statistical analysis accessible to researchers with minimal R or Python experience. Finally, by demonstrating the integration of statistical analysis results into molecular network visualization in Cytoscape, we offer researchers a method to enrich molecular network analyses with diverse informational layers, thereby enhancing the interpretability and impact of their findings from FBMN workflows.

Expertise needed to implement the protocol

We aim to make this protocol accessible to a broad range of researchers, from absolute beginners to experts. As we provide different options for executing the code (Web application, 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 Fig. 3. Furthermore, for a preview of the user interfaces across different platforms, Fig. 4a–c provides screenshots of a Jupyter Notebook, Google Colab and the web application. Recognizing that the documentation for each tool is distributed across various sources, we also include Fig. 4d, which outlines the locations where users can find comprehensive documentation for all the tools mentioned. At a minimum, we recommend having some general background in statistics and a basic understanding of LC–MS/MS data structure, as well as knowledge about the system and the experimental design of the dataset which should be analyzed. Furthermore, we strongly recommend consulting with a statistician to ensure accurate interpretation and application of the results.

Overview of the 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 have also developed workflows in Python and QIIME2 as well as a web application (details in Supplementary Information document). In the following sections, we provide step-by-step instructions of the R Jupyter Notebook. Code blocks are included to illustrate the main algorithms and functions.

The procedure is divided into five sections: 'Preliminary setup for the notebook', 'Data clean-up', 'Multivariate analysis', 'Univariate analysis' and 'Integrating statistical results into a molecular network'.

'Preliminary setup for the notebook' involves initial Steps 1–12, preparing the dataset for analysis.

'Data clean-up' covers Steps 13–30, focusing on blank removal, imputation, normalization and scaling. If blank removal was done in MZmine, skip Steps 21–23 and proceed directly to Step 24. The final step, Step 30, allows selection of a processed dataset, stored as '`cleaned_data`', for further analysis.

'Multivariate analysis' consists of the following sections:

- Steps 31–40 for PCoA and PERMANOVA, followed by hierarchical clustering (Steps 41–45) and heat maps (Steps 46–50)

- Step 38, which involves setting colors, is a prerequisite for executing Steps 47, 56, 64, 68, 71, 76 and 80, as all these steps require the color settings established in Step 38
- Clustering in Step 42 utilizes the distance matrix from Step 36. Also, Step 45 requires '`cleaned_data`' from Step 30. Hence it is recommended to perform clustering sequentially after PCoA and PERMANOVA
- Heat maps (Steps 46–50) use '`cleaned_data`' from Step 30 but are independent of previous multivariate tests
- RF classification (Steps 51–56) also relies on '`cleaned_data`' from Step 30 and is not dependent on the previous multivariate Steps 32–50

'Univariate analysis' (Steps 57–80):

- Other than Step 51, this section is not connected to the multivariate section
- The metadata from Step 51 is also essential for the 'Test for normality' (Steps 57–59), which informs whether to proceed with ANOVA and Tukey's post hoc (Steps 60–68) or KW and Dunn's post hoc (Steps 72–80). Note that Dunn's post hoc utilizes variables from KW and Tukey's from ANOVA
- The *t*-test (Steps 69–71) is an independent parametric analysis for two-group comparisons and can be performed after normality testing (Steps 57–59)
- While univariate analysis is not directly tied to multivariate steps, it is advisable to conduct at least one multivariate analysis, such as PCoA, to discern global patterns before continuing univariate analysis

The last step in the notebook, 'Exporting results', is detailed for Google Colab users, instructing on how to zip and download the session's result files. This step is unnecessary for Jupyter Notebook users as files are stored locally. Hence, it is only included in notebooks and not in the protocol.

'Integrating statistical results into a molecular network' section uses RF output from Step 56 as an example to integrate the results into the FBMN in Cytoscape.

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.

Materials

Software used

In our protocol, a variety of software options are offered to accommodate different user preferences and system capabilities. These include optional tools such as Google Colab for cloud-based operations, local installation of Jupyter Notebook, Streamlit web application and docker installation of QIIME2.

System configuration

The protocol's development and testing were conducted on a robust system, featuring an 11th Gen Intel Core(TM) i7-11700 processor with eight cores (16 CPUs) and a clock speed of 2.5 GHz, designed for efficient multitasking and rapid data processing. Complementing this, the system is equipped with 32 GB of RAM, enabling smooth handling of large datasets and a 1 TB solid-state drive (SSD), ensuring swift data access and processing. Operating under Windows 11 Pro, the system is compatible with all necessary software for the protocol. Additionally, it includes an Intel UHD Graphics 750 and an NVIDIA GeForce RTX 3070 GPU to meet a diverse range of graphical processing needs. The system is also connected to the university's Ethernet network, with a network speed of ~900 Mbps for both download and upload essential for efficient data transfer and cloud-based operations. The processing times outlined in the 'Procedure' section of this protocol are based on tests conducted with our example dataset on the Colab platform. Though the protocol is designed for compatibility with modern laptops or PCs, optimal performance may vary based on individual system specifications.

Table 3 | Sample metadata layout

Filename	ATTRIBUTE_groups	ATTRIBUTE_timepoint_hours
control_rep1.mzML	Control	1
⋮	⋮	⋮
Sample_type1_rep1.mzML	Sample_type1	4
Sample_type1_rep2.mzML	Sample_type1	4
⋮	⋮	⋮
blank.mzML	Blank	NA

The first column 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_hours", which is an example for numerical data.

Resources

For the feature quantification table, MZmine 3 (version 3.3.0) was used, installed on the computer with the above-mentioned system configurations and executed using a batch file, which is available along with all example input files in the Functional Metabolomics Lab GitHub Repository (https://github.com/Functional-Metabolomics-Lab/FBMN-STATS/tree/main/data). This batch file outlines each step taken in MZmine 3.

To cater to various user needs, the pipeline is accessible through several mediums. Google Colab offers a no-installation-required approach, ideal for those who prefer cloud-based solutions. The Streamlit web application (https://fbmn-statsguide.gnps2.org/) provides an intuitive GUI, suitable for users who favor visual interfaces. Here, users can directly upload all input files by simply entering the task ID from their FBMN job on GNPS. For larger datasets, or those seeking more control, a local installation of Jupyter Notebook is recommended. To assist users in choosing the most suitable method, a decision tree is provided in Fig. 3.

Recommendation for beginners and supplementary information

As a default for beginners, we recommend using the Colab Notebook with R code to follow this protocol. 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), with Supplementary Information detailing files beyond the R code.

Required files

Feature quantification table (.csv)

- Refer to Table 1 for an example. This table, typically generated from LC–MS/MS metabolomics studies, includes all mass spectral features (integrated peak areas) and their relative intensities across samples. While we used MZmine 3 to obtain the feature quantification table (in .csv format), users from other platforms, such as MS-DIAL, XCMS and OpenMS can integrate their tables by leveraging their FBMN task ID for seamless incorporation into the workflow. Manual data input requires referencing our dataset for necessary adjustments. See 'Resources' above for downloading the example feature quantification table (.csv) used throughout this protocol

Metadata (.txt)

- Refer to Table 3 for an example. The .txt file 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. See 'Resources' above for downloading the example metadata (.txt) file used throughout this protocol
- For the datasets to be fully considered for public meta-analysis, we suggest using a standardized metadata format with controlled vocabulary. For guidance, users can refer to the ReDU metadata template (https://mwang87.github.io/ReDU-MS2-Documentation/HowtoContribute/). Make sure the metadata format in this protocol

is 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_' (for example, ATTRIBUTE_groups, ATTRIBUTE_timepoint)
- Ensure metadata does not contain any empty cells for each of the attribute columns. Each cell must have a defined value. Fill all empty cells in the metadata table with 'NA'
- Make sure to include a fully populated column titled 'ATTRIBUTE_Sample_Type' to define various sample types (for example, blanks, samples, control and quality control (QC))
- In our example metadata, columns such as 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. Use consistent naming for the groups within ATTRIBUTE columns to avoid confusion and for ease of interpretation (for example, use 'control' consistently instead of variations such as 'control, control sample')
- 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 biological replicates (we suggest at least three) for each sample type within the experimental design

Additional input files

Besides the feature quantification table and metadata, the R and Python notebooks can also integrate molecular annotation files (in either .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 FBMN analysis. The process requires MS/MS fragmentation patterns in the '.mgf' format, a feature quantification table in '.csv' format, both obtained with for example, MZmine 3 (see 'Feature detection' section) and a metadata file in '.txt' or '.tsv' format. The .mgf file carries spectral information about specific MS/MS scans designated for each feature and feature identifications match with feature identifications in the feature quantification table. All 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[78], 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 at 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' subfolders (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 (for example, 100 Da), thus expanding the scope of annotation.

SIRIUS[53] can predict molecular formulas, as well as structures through structure database matching using CSI:FingerID[35,85]. Furthermore, the integrated CANOPUS[54] 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.

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 7 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 and Mission Bay, capturing both pre- (December 2017) and post-rainfall (January 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 (December 2017, January 2018 and October 2018) and two styrene-divinylbenzene polymer-based priority pollutant (PPL) solid-phase extraction process blanks at each of the sample times. The datasets can be found in the MassIVE repository: MSV000082312 and MSV000085786.

While metabolomics studies are typically focused on biological investigations and the example dataset is from an environmental system, the protocol is versatile and applicable to a broad spectrum of fields and sample types. This includes combinatorial chemistry, particularly in organic synthesis, doping analysis and trace contamination of food, pharmaceuticals and various other industrial products. It is also suitable for biological samples from diverse sources such as microbiomes, bioreactors or biomedical research. In general, the only requirement for the data to be processed using our protocol is compatibility with the feature table and metadata format specifications, making it a versatile tool for multiomics studies.

Seawater samples collected during October 2018 were not available in the original article yet. The .mzML files were preprocessed using MZmine 3 (version 3.3.0) (https://mzmine.github.io/) and the FBMN workflow can be openly accessed in GNPS (https://gnps.ucsd.edu/ProteoSAFe/status.jsp?task=b661d12ba88745639664988329c1363e). To review the specific parameters used for building this FBMN network, readers are encouraged to visit the provided GNPS link and the Supplementary Information document. Additionally, the Cytoscape files shown in the results can be found in our GitHub repository: https://github.com/Functional-Metabolomics-Lab/FBMN-STATS/tree/main/Integrating_Stats_Results_to_Molecular_Network.

Procedure

Preliminary setup for the notebook

▲ **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. Additionally, refer to Box 2 for general instructions on navigating the Jupyter Notebook, such as identifying code cells, recognizing those that require user input and adding comments.

1. Choose a notebook and install it. We recommend beginners 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 (for example, Intel i7, 16 core, 64 GB RAM) is typically faster and more efficient. The reported processing times here are based on our example dataset 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 install and run Jupyter Notebook locally in R, we have provided additional installation guides in the GitHub repository (https://github.com/Functional-Metabolomics-Lab/FBMN-STATS/blob/main/Jupyter-Notebook-Installation-Guides). Windows users are advised to use Anaconda Navigator for a straightforward setup, with detailed instructions provided in our guide. Extensive documentation for Anaconda Navigator can also be found at Anaconda's official site (https://docs.anaconda.com/free/navigator/tutorials/create-r-environment/). Mac OS users

BOX 2

General instructions for navigating the Jupyter Notebook

- Text in red: these sections indicate critical information or code 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: within the notebook, 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 cell. 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 code cell, at the notebook's top or even toward the upper section of your screen.
- Text in green: this indicates that the following code 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 code 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.

may encounter difficulties with Anaconda Navigator; thus, we suggest the Homebrew-based installation, detailed in the aforementioned link in our GitHub repository. For an introductory guide and additional tips on using Jupyter Notebook, please refer to Box 3.

Package installation

● **TIMING 15 min**

2. Install the R packages:
   - The notebook utilizes R version 4.1.3 (10 March 2022)
   - Begin by installing and loading the necessary R packages using the `p_load()` function from the 'pacman' (v0.5.1) package[86]. 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.

   The following R packages are essential for this protocol:
   - Data clean-up: tidyverse[87] (v2.0.0), IRdisplay[88] (v1.1), KODAMA[89] (v2.4)
   - Multivariate statistics: factoextra[90] (v1.0.7), vegan[91] (v2.6-4), ComplexHeatmap[92] (v2.10.0), dendextend[93] (v1.17.1), NbClust[94] (v3.0.1), rfPermute[95] (v2.5.1)
   - Univariate statistics: FSA[96] (v0.9.4), matrixStats[97] (v0.63.0)
   - Visualization and plotting: ggsci[98] (v3.0.0), cowplot[99] (v1.1.1), svglite[100] (v2.1.1)

   At this stage, install the packages required for data clean-up.
   ▲ **CRITICAL STEP** Packages are installed just before their respective sections (in Steps 2, 26, 31, 51, 57) 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 clean-up.
3. 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

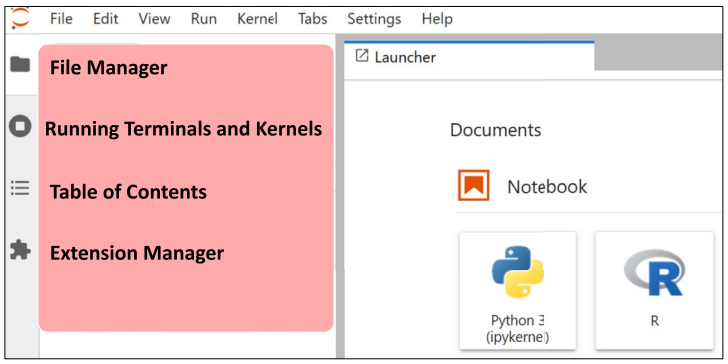
BOX 3

Using JupyterLab to manage Jupyter Notebooks locally

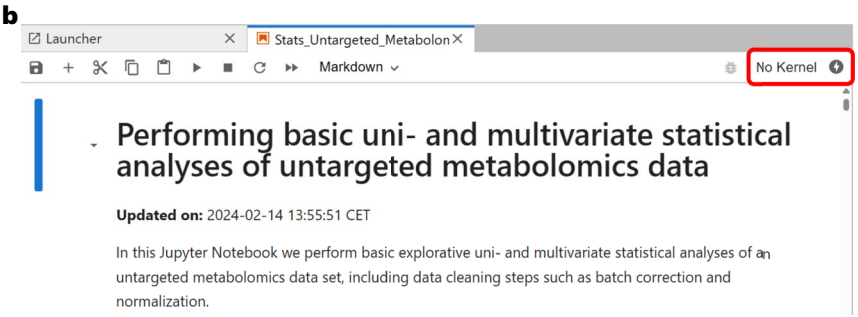
JupyterLab serves as an integrated development environment for managing Jupyter Notebooks. It offers a user-friendly way of working with multiple notebooks simultaneously, sharing variables between notebooks and saving notebooks mid-session so that analysis can be continued at a later time.

Getting started: For those new to running Jupyter Notebooks locally, we have included a section on how to install and open JupyterLab in our installation guidelines. The section also includes a short introduction to the JupyterLab interface.

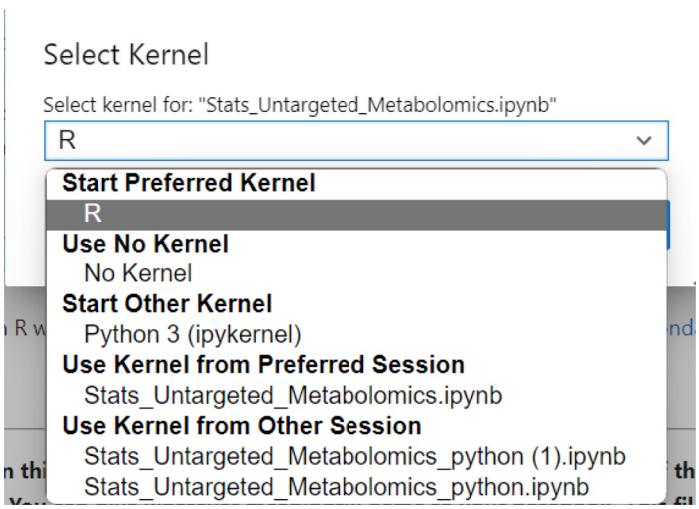
a



b



c



(continued from previous page)

Navigating JupyterLab: Upon launching JupyterLab a workspace will open as shown in panel **a**.

Here, the left sidebar features four main icons: (1) file browser: access and manage your files; (2) running terminals and kernels: view your active sessions; (3) table of contents: quickly navigate through the sections of your current notebook. This is particularly useful for large notebooks; (4) extension manager: customize JupyterLab with additional features.

Opening a Jupyter Notebook locally: panel **b** shows the view of an opened Jupyter Notebook. There are two options to open a notebook locally: (1) navigate to the location of your notebook using the file browser and double-clicking the notebook or (2) Go to 'File → Open from Path' and enter the path to your notebook (for example, `Downloads/Stats_Untargeted_Metabolomics_python.ipynb`).

Selecting a kernel: selecting the correct kernel is important for successful execution of a notebook. The kernel active in the current notebook is displayed in the top-right corner (red box in panel **b**). The gray circle indicates the state of the kernel (unfilled = idle, filled = running).

Click the name of the current kernel to start or switch between kernels. A dropdown menu will pop up as shown in panel **c**.

'Start preferred kernel' and 'start other kernel' will start a new kernel for the notebook. For R notebooks, make sure to select the R kernel. If you have already started a kernel for another analysis and wish to carry over all variables and settings to the current analysis, choose a kernel from 'use kernel from preferred session' or 'use kernel from other session'.

Saving your work: navigate to the 'File' tab found in the top left corner of the interface. Here, two main options are available to save your work: (1) save notebook: saves the state of your current notebook, does not preserve variables. (2) save workspace: saves the current notebook and all variables to avoid having to rerun the entire notebook upon return.

Further information: for detailed installation instructions and further guidance, please refer to our installation guidelines over on GitHub (https://github.com/Functional-Metabolomics-Lab/FBMN-STATS/blob/main/Jupyter-Notebook-Installation-Guides) and the official Jupyter documentation (https://docs.jupyter.org/en/latest/).

- 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'. Further details on using Google Colab are provided in Step 4, along with Fig. 5 for visual guidance
- To set your working directory in the notebook, run the code cell (a cell or a section containing codes) as shown in Fig. 6. A pop-up box will appear as illustrated in Fig. 6, prompting you to enter the path to your designated folder with input files. Insert the respective path and press 'Enter' to proceed to the next cell
- (For local environment) If you are 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'

Data import

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

4. Uploading files to Google Colab (user input required). Some general guidelines for using Google Colab are as follows:

   Kernel selection: When setting up the Google Colab environment, ensure you have the correct kernel (R or Python) activated for the notebook. To do this, go to the 'Runtime' tab, and select 'Change runtime type' to choose your desired kernel. For multiple R versions, such as 'R' and 'R.4.2.2', we suggest choosing 'R' to ensure compatibility.

   Interface navigation: The Google Colab interface is user-friendly, featuring some useful icons on the left sidebar (Fig. 5a). The 'table of contents' icon facilitates quick navigation through the notebook and a 'find and replace' icon for efficient content search. Managing your files is streamlined through the 'files' icon, where uploading and organizing data is straightforward.

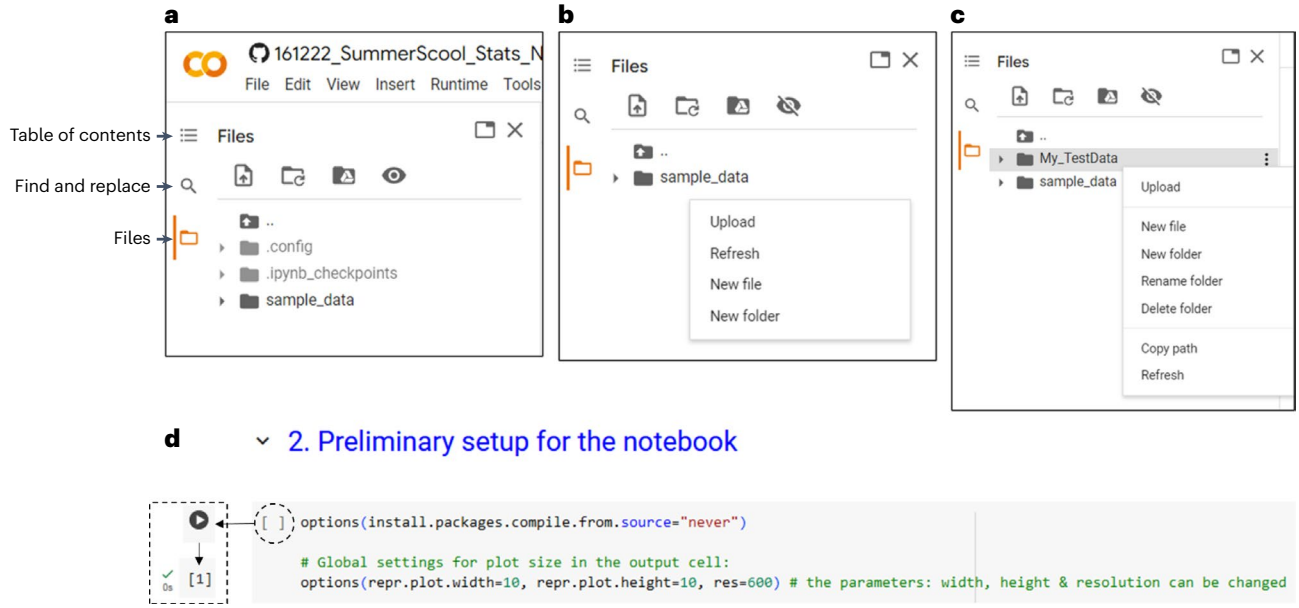


Fig. 5 | Google Colab interface for managing R notebooks. a, Menu and left sidebar highlighting icons, 'table of contents', 'find and replace' and 'files', with the 'files' icon selected in orange. **b**, The action of right-clicking in the 'files' area shows options for new folder creation, alongside the default 'sample_data' folder in the Colab. **c**, User-created 'My_Testdata' folder for organized data storage, demonstrating file upload options. **d**, Showcases the code execution process with empty square brackets before running, changing to a play button upon hover and displaying execution order ('1') and time (0 s) after running.

File management:

- Uploading: Google Colab does not allow local file access. Hence, directly upload files from your computer to Google Colab. If you do not want to use files from your local machine, you can skip this (Step 4) and proceed directly to Step 5B ('loading files from URL') or Step 5C ('loading files from GNPS')
- Organization: for an orderly workspace and clear data arrangement, create folders for your dataset. Right-click in the 'Files' area to upload data, refresh the view or create new folders (Fig. 5b). Figure 5c shows the folder 'My_TestData', created for organizing user data. Right-click on the created folder and select 'upload' to transfer files from the local machine to the cloud session. Alternatively, one can also 'drag and drop' files directly into the folder
- Google Drive access: linking Google Drive is simpler with Python. It is less direct with R, hence not advisable

Executing code:

- Google Colab distinguishes code cells (in gray) from text annotations, also called 'markdown cells' (in white). Run the code by clicking the play button on each cell's top left corner (Fig. 5d)
- Execution outcomes are indicated by a green check (for success) or an error message (for failures), providing immediate feedback (Fig. 5d)

Fig. 6 | 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 (for example, /content/test_data) and press enter.

- One can also run several code cells sequentially within the notebook. Look for an arrow icon next to the section titles in the notebook (refer to Fig. 5d, illustrating section header '2. Preliminary setup for the notebook'). Clicking this icon reveals the total number of concealed cells within that section, such as '84 hidden cells', including both code and markdown cells. Activating the play button situated below the section title initiates the execution of all these cells in a collective manner. However, it is important to be cautious with these batch runs, especially if you are not familiar with the steps being executed, as some may require user input. Batch runs are most useful for sections that do not need any input from the user

5. Select a data loading method. There are three options. Files can be uploaded (A) from a working folder, (B) from a URL or (C) directly from a repository, for example, GNPS.
   (A) **Loading files from a folder**
   (User input required)
   (i) Begin by viewing a table displaying a list of files in your working folder (for example, uploaded in the previous step) as shown in Fig. 7a. Each file will have an index number associated with it. Your task will be to import three tables by specifying the index number associated with each: the feature quantification table (`ft`), the metadata table (`md`) and optionally, annotation tables (`an`) (see Fig. 7b for an example). To guide you through this process, there are three code blocks that require user input.
   (ii) 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.
   (iii) 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.
   (iv) SIRIUS annotation file import (optional): the third code block in the notebook queries if you have an additional SIRIUS annotation file (Y/N). If 'Y', you will enter the file's index number; if 'N', you move to the next cell. This file will be used to merge all annotations (for example, GNPS library, analog hits and SIRIUS) into a single master table for easier data exploration later on. It is worth noting that this

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
```

Fig. 7 | 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, where '5' corresponds to the feature table file 'SD_BeachSurvey_GapFilled_quant.csv' and '2' to the metadata file "20221125_Metadata_SD_Beaches_with_injection_order.txt", as shown in **a**.

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.

▲ **CRITICAL STEP** 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.

(B) **Loading files from a URL**
(User input required)

(i) We also provide an example of retrieving data from a URL. For this example, open the following URL to obtain the feature quantification table: https://raw.githubusercontent.com/Functional-Metabolomics-Lab/FBMN-STATS/main/data/SD_BeachSurvey_GapFilled_quant.csv).

(ii) Access the feature quantification table, metadata and analog result files directly from our GitHub page (https://github.com/Functional-Metabolomics-Lab/FBMN-STATS/tree/main/data).

▲ **CRITICAL STEP** The files can be sourced from any public site where your data is stored, such as GitHub. If you have your dataset hosted on GitHub or a similar platform, simply use the file's URL.

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.

(C) **Loading files from GNPS**
(User input required)

(i) 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.

(ii) 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.

Exploring the imported files

6. 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, that is, 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.
7. 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.
   - INDEX: row number in the summary table
   - ATTRIBUTES: column name of the attribute, for example, ATTRIBUTE_Sample_Type
   - LEVELS: unique groups within the attribute column, for example, blanks, sample
   - COUNT: number of files for each category, for example, 6, 180 indicating 6 files for 'blank' sample type and 180 for 'sample' sample type
   - ATTRIBUTE_CLASS: data type of the attribute. Useful for spotting cases where a numeric attribute like ATTRIBUTE_minutes is classified as 'character'

Merging annotations with feature quantification table

▲ **CRITICAL** 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.

8. Identify 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: similarly 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
9. Ensure data compatibility before merging. 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.
10. Merge 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

11. (Optional) Incorporate additional annotations:
    - 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`' dataframe based on '`row ID`'
    - For simplicity, we have shown here how to merge the summary file from the SIRIUS module. Given the versatility of SIRIUS, which includes other modules like ZODIAC for molecular formula predictions, CSI: FingerID for molecular structure prediction and CANOPUS for compound class prediction, users will obtain separate summary files for each module. Our protocol shows one example of merging summary files from the SIRIUS module, the commonly used module, storing the output in a variable named 'sirius'. Users working with summary files from CANOPUS or other modules can similarly save and adapt the code using a different variable, like 'canopus', allowing for flexible adaptation as per their requirements
12. Export the merged annotations. To do this, write the merged annotation table to a CSV file for further data exploration and downstream analyses.

Ensuring metadata and feature quantification table compatibility for downstream analysis

▲ **CRITICAL** 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 in the Jupyter Notebook. Simply run each code cell in turn.

13. Create 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.
14. Clean 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
15. Update 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
16. Select 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 to the original feature quantification table
17. 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 clean-up
    - 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. Rerun all the above steps once corrected

Data cleaning

● **TIMING 20–30 min**

▲ **CRITICAL** Following the LC–MS/MS data preprocessing with MZmine, we perform the postprocessing 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.

18. Transposing the feature quantification table:
    - No user input is required other than running this code cell. The transposition and all subsequent actions will be executed automatically. 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

(Optional) Batch correction

▲ **CRITICAL** The most common method for visualizing or identifying the presence of batch effects is through a simple PCA, guided PCA[101] or PCoA. In the PCA/PCoA scores plot, it is generally expected that all the (pooled) QCs cluster together indicating little analytical variation in the data. When the interbatch variation gets higher, the inter-QC distances in the PCA/PCoA plot will also increase[102].

19. Batch effects can be detected and corrected using either the steps outlined in this notebook (option A) or established literature methods (option B). Option A involves steps that refer to PCoA, detailed later in the procedure. We have deferred the resulting visualization to a later section, after data clean-up. As we delve deeper into multivariate analyses after data clean-up, this approach avoids redundancy and ensures users can maximize the utility of this protocol.
    (A) **Using the notebook:**
        (i) Execute Step 31 to install the necessary packages for multivariate analysis.
        (ii) Run Steps 38 and 39 to visualize the PCoA using the custom-made 'plotPCoA()' function. Detailed usage instructions are provided in the respective steps.
        (iii) If your sample type information (description of which samples are pooled QCs, blanks, samples and so on) is located in the 'ATTRIBUTE_Sample_Type' column of the metadata, invoke the function by typing:

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

    (B) **Established literature methods:**
        (i) Determine whether there is a batch effect using ANOVA by comparing the QC mean of different batches for statistically significant differences[103].
        (ii) If there is a notable batch effect, choose an appropriate approach to correct the effects. These include:
            - Normalization methods such as Metabodrift[104], ComBat[105]
            - Transformation method: waveICA[106] (wavelet transformation coupled to ICA)
            - Regression-based approaches such as the linear least-square method[107], QC-based robust LOESS correction[108], QC-support vector regression[109]
            - ML-based methods such as RF-based QC-RFSC correction[110], deep learning model: normalization autoencoder algorithm[111], regularized adversarial learning preserving similarity[112]

Each method has its strengths and limitations. It is important to note that, depending on the experimental design (for example, sample size and sample diversity), pooled QCs are not always utilized. In situations where there are no QCs included in the study, normalization can be used instead to attempt to reduce most of the unwanted variations[113], 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

interbatch correction similarly to the method described by Qin Liu et al.[114]. The procedure involves calculating the mean of each feature across all batches, then calculating the batch-specific feature means and subsequently adjusting feature intensities within each batch relative to the batch-specific and overall means. For intrabatch adjustments, this additional notebook illustrates the QC-based LOESS correction method, with a prerequisite that each batch should start and end with a pooled QC injection.

Blank removal

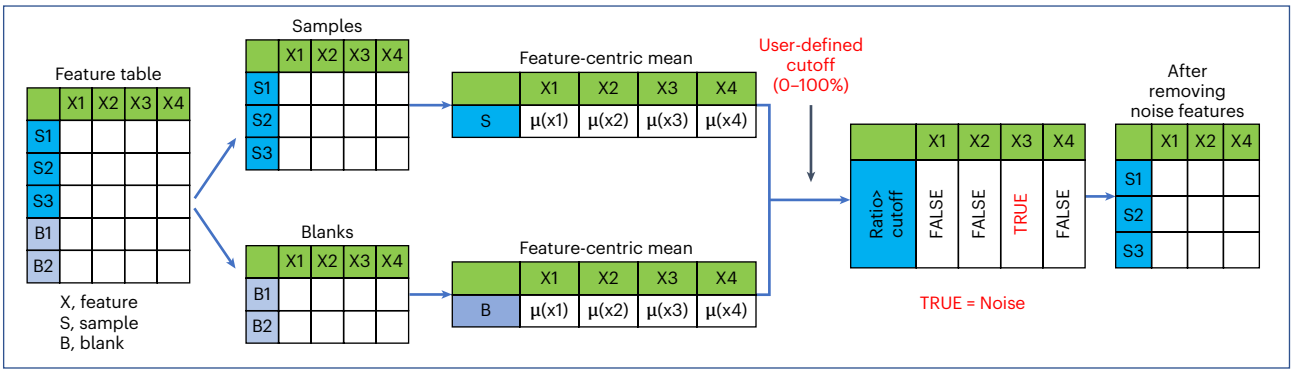
▲ **CRITICAL** To prioritize or identify metabolites from our samples, we need to remove contaminants, that is, features found in the blanks, before proceeding with statistical analysis[115]. While blank removal can be executed during preprocessing with MZmine 3, which might result in the absence of blank features and samples in both the feature table and metadata, conducting it during postprocessing offers more flexibility (see Fig. 8 for a graphical overview of blank removal and refer to Box 4 for more insights).

▲ **CRITICAL STEP** If you have performed blank removal during preprocessing, simply skip Steps 21–23. Before Step 24, assign your feature table and metadata to '`blk_rem`' and '`md_Samples`', respectively, as in the following code block. This step prepares the data for seamless integration into subsequent steps.

```
blk_rem <- ft_t
md_Samples <- new_md
```

20. 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 (for example, 'blanks', 'samples').
21. Separating blank and sample files (user input required). In this step, the data is split into two groups: 'blank' and 'sample' files. It is important to note that 'samples' here include all mzML (or mzXML) files except blanks, including control samples, as they might be also 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' and so on
    - 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 (for example, '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 (for example, 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
22. 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.

    ▲ **CRITICAL STEP** 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.




```r
#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"
```

```r
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

Fig. 8 | Blank removal process. Featuring a graphical representation of the blank removal followed by screenshots of the corresponding R code executed for the procedure.

23. 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`'

BOX 4

Blank removal

To obtain reliable and meaningful LC–MS/MS metabolomics data, it is crucial to integrate QC samples and blanks throughout the measurement process, which can facilitate blank removal. This step is critical to eliminate background noise such as signals from plasticizers, solvent impurities or sample clean-up reagents, along with cumulative carryover contamination[145,149]. Therefore, incorporating blanks during sample collection, preparation, and measurement is vital for identifying and eliminating these interferences, as detailed in Box 1.

Removing these noninformative features improves the quality and interpretability of the data by reducing the dataset's dimension and minimizing false correlations. Nevertheless, one should be aware of potential challenges, such as system deconditioning, which can lead to systematic variations in the metabolomic profiles. To counteract this, careful placement of blanks within a sample batch is advised to minimize such artifacts and to maintain data integrity[149].

One of the existing methods to remove the noninformative features is creating a molecular network using the online platform, the GNPS and visualizing the network in Cytoscape. While this process is reliable, it is tedious and requires users to manually remove blank and media nodes from the molecular network.[7]. There is also the 'msPurity' R package from Lawson et al.[150] with a function called 'SubtractMZ' to perform blank removal. Data-adaptive filtering methods have also been suggested to remove features from blanks and low abundant features from samples with undetected values[151].

Another popular feature filtering method is based on the coefficient of variance (CV). Also referred to as relative standard deviation is a measure of statistical dispersion, calculated as the ratio of the standard deviation to the mean[152]. 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 Food and Drug Administration studies. However, it is essential to approach CV filtering with caution. Schiffman et al.[151] 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[151]. 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.[149], 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[149].

- 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 notably present in blanks as compared with 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 notably 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 are also displayed. The code block used in the notebook for this Step 23 is shown in Fig. 8.

Imputation

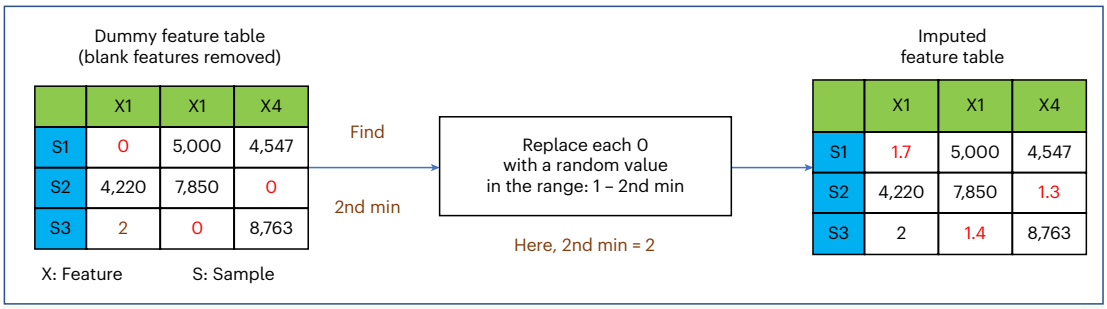
▲ **CRITICAL** Many feature extraction software programs, such as MZmine 3, often generate tables with missing values denoted as 'NA', 'NaN' or 0. This means that for several *m*/*z* and RT traces in a given sample, there may not be a peak detected and therefore no value is available[83]. However, many statistical approaches, such as 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.

BOX 5

Imputation strategies

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

1. Below the limit of detection: if a value is missing because the corresponding molecule was below the analytical method's limit of detection, consider replacing missing values with a low value, ensuring it does not artificially lower the variance[153]. Our imputation method corresponds to this scenario.
2. Sample processing or feature extraction artifacts: missing values due to sample processing or extraction issues, such as ion suppression or retention time shifts, may prevent accurate




```
#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"
```

```
# 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])
}
```

(continued from previous page)

peak detection for certain *m/z* and RT traces. Furthermore, matrix effects may complicate metabolite quantification, leading to data gaps despite the presence of peaks in the raw data[83,154]. For comprehensive statistical analysis, one can consider imputing missing values with those similar to values detected in other samples. Here, machine learning methods such as KNN or RF can be useful. KNN fills in multiple missing values by identifying the *k* nearest data points to a given point[154]. Similarly, RF can iteratively impute missing values using a proximity matrix derived from RF classification across all metabolites[83].

However, caution is advised with imputation as it introduces data points where none existed, potentially skewing results. One needs to be aware of the risks and limitations of imputation in statistical analysis.

A visual representation of the imputation algorithm complemented by screenshots of associated R code snippets is shown below.

Refer to Box 5 for more information on imputation strategies and see the accompanied illustration for a graphical overview of imputation.

▲ **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.

24. Analyzing the frequency distribution of relative intensities. Examine the frequency distribution of the relative intensities in the 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 1,000.
25. Replacing zeros with random values. The program replaces all zero values in the dataset with the randomly generated number between 1 and the smallest nonzero 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. For the rationale behind this approach, see the first point in Box 5.

Normalization

▲ **CRITICAL** Sample normalization aims to eliminate systematic bias via adjusting variations across samples[116]. 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 26–29), but they can choose either method for further analysis in Step 30. 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 6. For a graphical view of the provided normalization methods, see the accompanying illustration in Box 6.

26. Install the KODAMA package.
27. Run the TIC normalization method. This step does not require any user input other than running the code cell. In TIC normalization, also known as total sum normalization, every feature within a sample is normalized relative to the area of the TIC chromatogram[117]. This involves dividing each feature by the sum of the peak areas of all features within a sample. The normalization function from the KODAMA[89] (v2.4) package performs row-wise sum operations; for this to work, the sample names are arranged in rows and their features in columns.
28. Run the PQN method. The user can simply execute this code cell without needing to input anything. 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[118]:
    - Normalization of test spectrum: the test spectrum is first normalized, typically using a sum normalization technique such as 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 QC samples

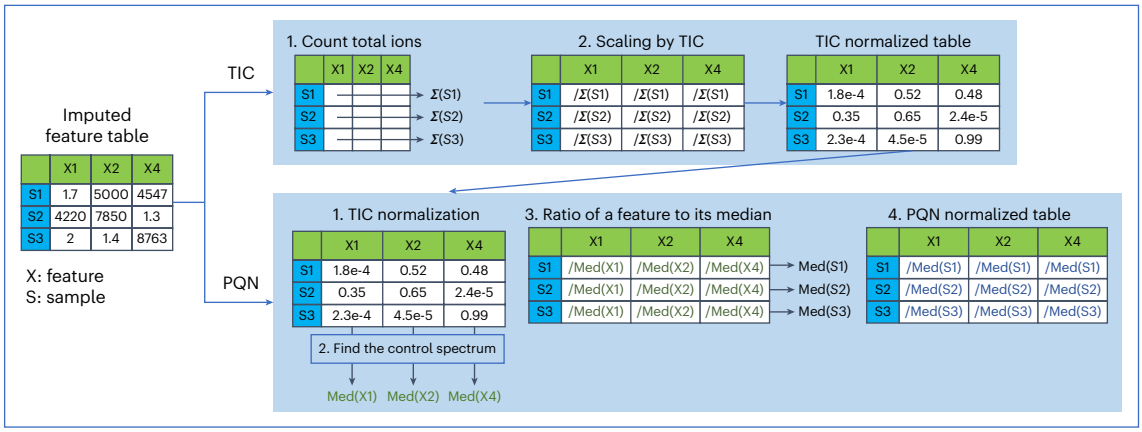
BOX 6

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[116,155]. 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[156] 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 substantially across different samples or experimental conditions and their effectiveness decreases with large global changes in metabolite levels (for example, 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 diseased conditions[157].
2. PQN[118]: Recommended when notable size effects are present or when internal normalization disrupts relative peak information[116]. Among several LC/MS-based normalization methods, including Contrast Normalization, Cubic Splines, and Cyclic Loess, PQN has been identified as the best performer in reducing sample-to-sample variations[155].
3. Common components and specific weights analysis[158] (CCSWA): a viable alternative when QC and sample data differ.

A graphical representation of TIC and PQN methods, accompanied by corresponding R code snippets are shown below.




```
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)))
```

- 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

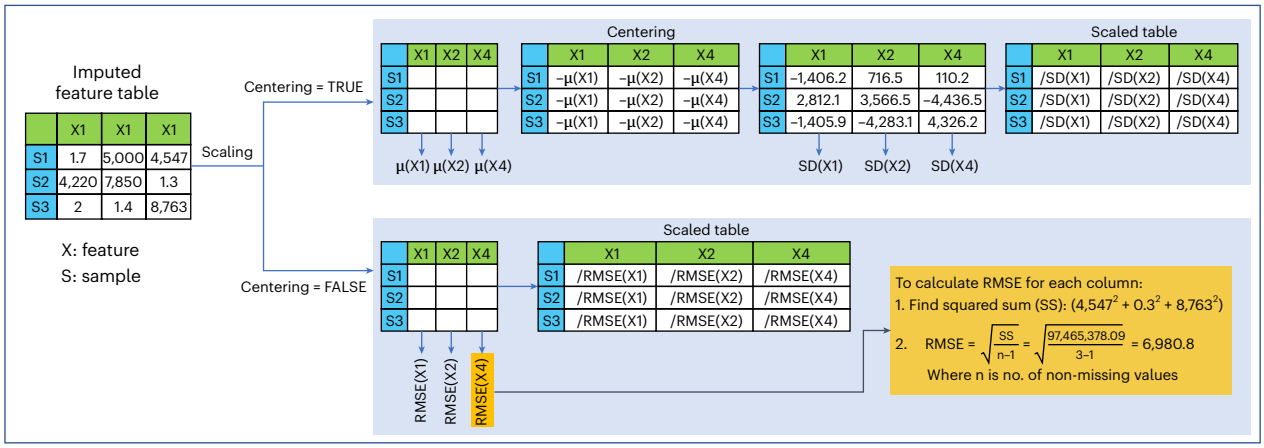
BOX 7

Scaling

Scaling methods can be categorized into two subclasses based on the scaling factor used[119].

1. Using data dispersion methods, such as standard deviation (s.d.), for scaling: examples include: autoscaling[159] and Pareto scaling[160]. Autoscaling ensures equal variance (such as s.d. = 1) for each variable, while Pareto scaling uses the square root of s.d. 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[119,161].

A graphical representation of the auto-scaling method, both centered and non-centered approaches, accompanied by corresponding R code snippet from the R notebook using the 'scale' function is shown below. Centering adjusts the data such as the fluctuations are centered around zero rather than the mean of the metabolite concentrations[119]



the test spectrum values relative to the control spectrum, ensuring an equal basis for comparison across all samples

Scaling

▲ **CRITICAL** Scaling methods in metabolomics aim to adjust the range of peak abundances between features[116]. This is done by normalizing the intensities of each feature by a scaling factor, effectively adjusting for fold differences between features[119]. Additional information on scaling factors can be found in Box 7 along with the graphical representation of scaling.

29. Run the center-scaling method. Simply run the code cell. The program applies 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[119]
    - 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 STEP** Since scaling introduces negative values, trying a PCoA with the Bray–Curtis difference on scaled data will trigger an error.

30. Choosing data for further analysis (user input required). Upon executing this step, an overview table is generated automatically, offering a list of the dataframes produced during each phase of data clean-up along with its respective metadata tables. This includes stages such as the initial raw data (raw data), postblank removal data (blank removed data), postimputation data (imputed data) and various normalization stages (TIC normalized, PQN normalized, scaled data). To proceed after the overview table is generated, the user should select a 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

    Two types of analyses are meaningful:
    - 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 such as PCA, it is important to ensure that a few dominant features do not skew the overall results
    - Using scaled data in multivariate techniques such as PCA prevents high variance features from dominating. Additionally, techniques relying on distance measures, such as *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 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.

Multivariate statistics

● **TIMING 50–60 min**

▲ **CRITICAL** In this section, we describe four approaches to mulitvariate analysis. We expect that users will choose one or more of these based on the information provided in the introduction.

- PCoA with PERMANOVA
- HCA
- Heat maps
- Supervised classification: RF

31. To start the multivariate analysis, install and load the necessary R packages for this section: BiocManager[120] (v1.30.9), ComplexHeatmap[92] (v2.10.0), dendextend[93] (v1.17.1), NbClust[94] (v3.0.1), ggsci[98] (v3.0.0) and cowplot[99] (v1.1.1). This process takes 5–10 min.

PCoA with PERMANOVA: PCoA

▲ **CRITICAL** Refer to Box 8 for a detailed overview of PCoA, including a graphical illustration of the PCoA process and corresponding code snippets used to perform the following steps in the notebook.

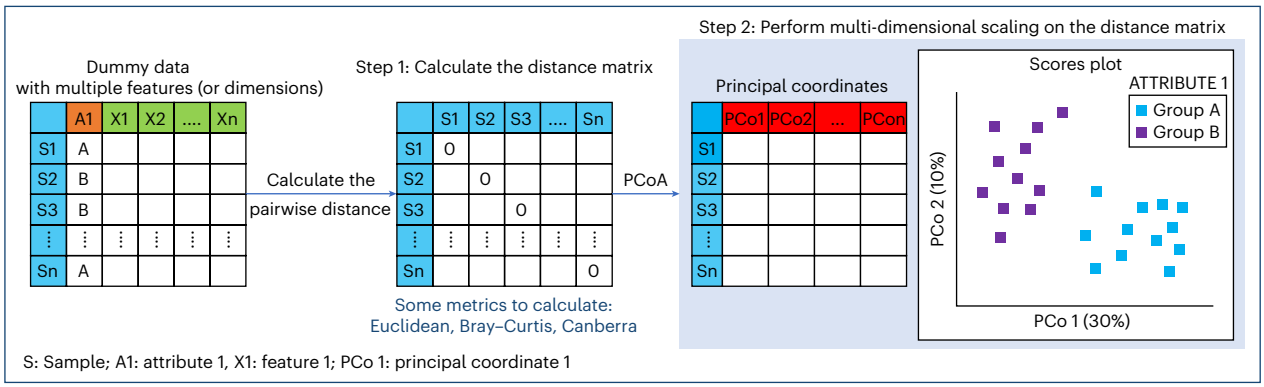
BOX 8

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[61,162,163]. Akin to phylogenetic distance measures such as UniFrac distance[164] used in the microbial ecology field, chemical distance matrices are emerging that make use of cosine MS/MS similarity between features[165] or chemical similarity derived from CSI:FingerID[166].

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 PCos, unlike PCA which offers 'loadings' information[167]. To discern associated features in such contexts, it is recommended to complement PCoA with other methods like a heat map overview, RF 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[168]. 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 on the basis of 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. The diagram below illustrates the process of transforming feature quantification tables into score plots by calculating distance matrices and plotting PCos. The associated code demonstrates MDS using Euclidean distance. Notably, using Euclidean with PCoA is the same as performing PCA; however, the users can adjust to other metrics, such as Canberra.



S: Sample; A1: attribute 1, X1: feature 1; PCo 1: principal coordinate 1

```
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

32. Prepare data. This step makes sure that the metadata (‘`metadata`’) and the feature quantification table (‘`cleaned_data`’) are in the same order. Also, we verify that the sample names (row names) in both data tables are identical and in the same order using the `identical()` function. It should return TRUE.
33. Calculate pairwise distances and perform PCoA. No user input is required. We calculate pairwise Euclidean distances across all samples in the feature quantification table using the `vegdist()` function from the ‘vegan’ package[88].
    - 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 STEP** The `vegdist()` function offers various methods such as ‘manhattan’, ‘euclidean’, ‘canberra’, ‘bray’, ‘jaccard’, ‘gower’, ‘binomial’ and ‘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 the ‘cleaned_data’ we use is already centered and scaled, we set these parameters to FALSE. For loadings and PC scores, you can access ‘`pca_result$rotation`’ and ‘`pca_result$x`’ respectively.
34. Analyze PCoA results. No user input is required in this code cell. The user can 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
35. Plot PCoA scores (user input required). Using the ‘ggplot2’ library, create a PCoA scores plot. Here, the samples are color-coded on the basis of the ‘ATTRIBUTE_Month’ attribute. To view the sample distribution of different attributes, the user can 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[59].

PERMANOVA

▲ **CRITICAL** Before performing PERMANOVA, it is important to validate the homogeneity of group dispersions, often termed ‘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)[91,121]. 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 9. For a visual representation of assessing multivariate dispersion and conducting the PERMANOVA analysis in R, refer to the accompanying code snippets in Box 9.

36. Test for homoscedasticity (user input required):
    - 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`’
    - Similarly to Step 33, 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, evaluate group dispersion against the chosen attribute group
    - Visualize the dispersion model to offer a clearer perspective

BOX 9

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 such as 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[121].

Panel **a** is the R code snippet for testing multivariate dispersions within the 'group', specifically referencing the 'ATTRIBUTE_Month' column (December, January, October) from the metadata.

Similarly, the R code snippet for executing PERMANOVA to analyze variations between the aforementioned groups in the previous illustration is in panel **b**.

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

- Perform ANOVA on the dispersion model. A significant *P* value ($P < 0.05$) indicates a violation of PERMANOVA's foundational assumptions. Conversely, a nonsignificant result suggests that PERMANOVA is a suitable choice for the given attribute
  ▲ **CRITICAL STEP** The resulting *P* value for 'ATTRIBUTE_Month' indicates the presence of group dispersions, in this example the group dispersions are among different months. This violates the PERMANOVA assumption. When PERMANOVA is performed for this attribute, the PERMANOVA results require a more cautious interpretation.

37. Conduct the PERMANOVA test. To do this:
    - Use the `adonis2()` function from the 'vegan' package 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
38. Define a function for streamlined analysis. To facilitate quicker analysis and avoid rewriting from Step 33 to Step 37 for testing different parameters, we defined a function, `plotPCoA()`. This function performs a PCoA using a chosen distance metric, calculates PERMANOVA and plots the results in a two-dimensional graph. Additionally, it assesses group dispersion before the PERMANOVA calculation and displays the significant 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

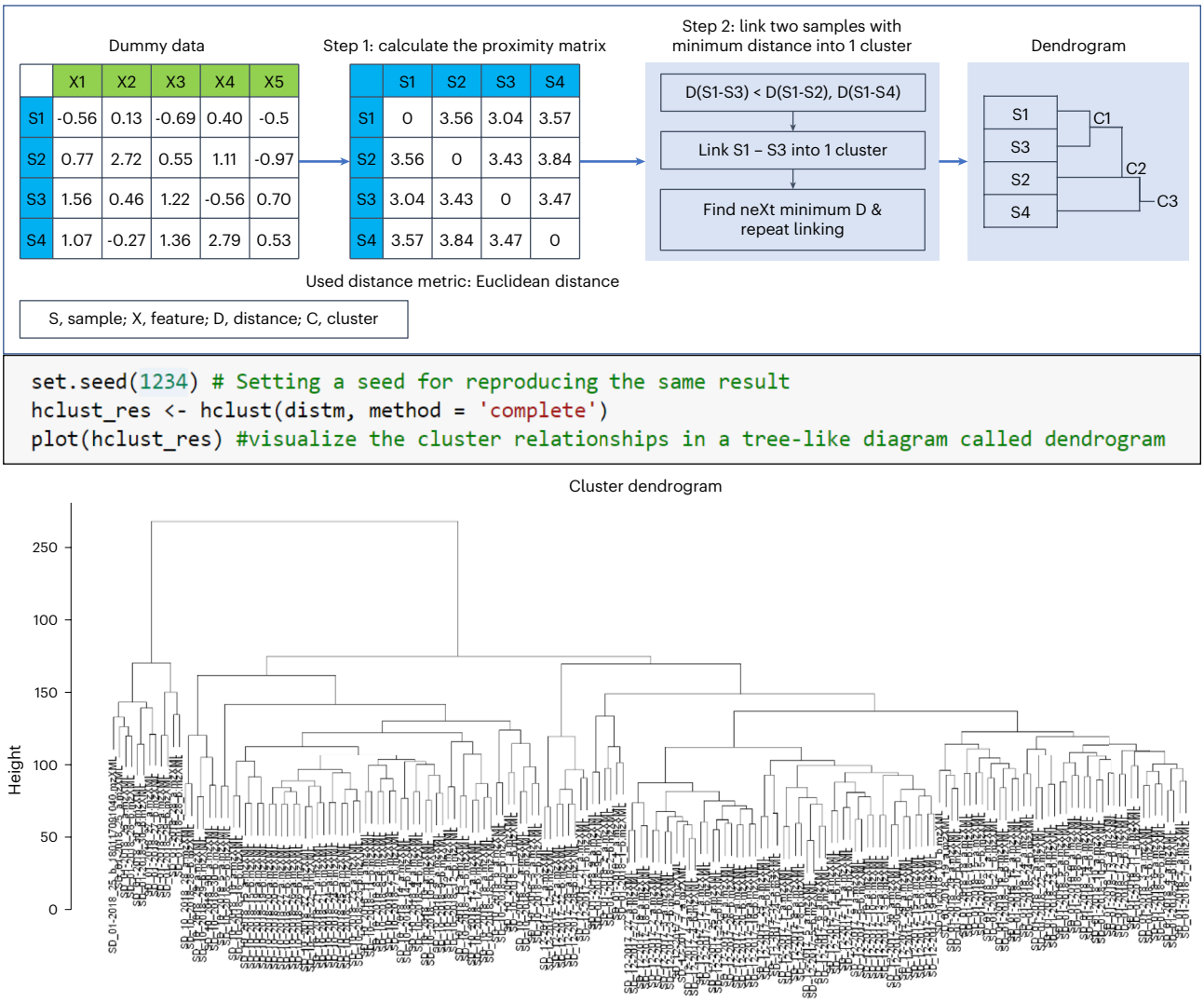
    Furthermore, we have created a '`custom_palette`' of 22 colorblind-friendly colors for coloring the groups consistently across the protocol. This color palette is used in Steps 47, 56, 64, 68, 71, 76 and 80. Users can customize this palette in Step 38 according to their preferences. Additionally, we 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 used after each plot creation to save the visualizations.
39. Apply `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')
```

40. Get PCoA plots after each data clean-up step (user input required). Specify parameters, such as the distance metric, attribute for PERMANOVA calculation, attribute to color the PCoA scores and the category of the chosen attribute, similarly to the previous plotPCoA step. These inputs will be taken to produce an overview of PCoA plots for all steps of data clean-up.

Hierarchical cluster analysis (HCA)

41. Set the plot size. It is important to define the size of the output plot, as dendrograms are typically larger in plot size. Adjust the plot size accordingly to ensure a clear and comprehensible visualization.



	X1	X2	X3	X4	X5
S1	-0.56	0.13	-0.69	0.40	-0.5
S2	0.77	2.72	0.55	1.11	-0.97
S3	1.56	0.46	1.22	-0.56	0.70
S4	1.07	-0.27	1.36	2.79	0.53

	S1	S2	S3	S4
S1	0	3.56	3.04	3.57
S2	3.56	0	3.43	3.84
S3	3.04	3.43	0	3.47
S4	3.57	3.84	3.47	0

Fig. 9 | Dendrogram generation and analysis. A dendrogram as a result of applying HCA to a feature quantification table (for example, '`cleaned_data`'). From this data, a proximity matrix (or the distance matrix) is calculated (Steps 33 and 36), 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 36, 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 subclusters within these main clusters are also discernible.

42. Execute HCA. No user input required. Here, we use the `hclust()` function from the 'stats' package to perform HCA. The function is applied to the distance matrix '`distm`', calculated on the basis of the feature quantification table ('`cleaned_data`') using a specified distance metric (for example, Euclidean, Canberra). The '`method`' argument in `hclust()` denotes the linkage method used for measuring the distance between clusters (for example, complete, single and 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 as shown in Fig. 9. This dendrogram shows split or merge distances as 'height' along the *y* axis, providing a visual representation of the cluster formation.
43. Cut the dendrogram (user input—optional). Similarly 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. Refer to Step 45 for more details on choosing the number of clusters.

44. Color 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.
45. Determine 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[71]
    - 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[71]. However, the 'elbow' point is considered an indicator of the optimal number of clusters, as further cluster additions do not notably improve the clustering or decrease the WSS. For our example data, this method suggests three or four clusters. However, defining the 'elbow' can be subjective
    - An alternative 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. However, there are more sophisticated techniques such as 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[71].

▲ **CRITICAL STEP** All of the heuristic methods in Step 45 serve as guidelines rather than definitive answers. In practice, in Step 43, 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. While context-based clustering might provide initial insights, it is important to avoid biases that could arise from relying heavily on pre-existing knowledge or expectations. An objective analysis where the data substantiate the clustering choice is crucial to ensure that the results are reflective of true patterns in the data, rather than what one expects or wants to see.

Heat maps

▲ **CRITICAL** Heat maps are generally used to visualize complex data or discern patterns across a high-dimensional dataset. They are commonly used in bioinformatics[122], particularly in gene expression analysis and visualizing genomic datasets, owing to their ability to effectively represent thousands of data points[123]. This makes them equally suitable for MS-based metabolomic experiments. Heat maps are efficient in pattern recognition due to their color-coded matrix elements and adjacent dendrograms, which indicate functional relationships between variables and samples[72]. To see the resulting heat map generated by the R code in the notebook, refer to Fig. 10. In this section, we will show how to incorporate hierarchical clustering into our heat map.

46. Preparing metadata for the heat map (user input required). To start with, determine which metadata columns or attributes will be used to decorate the heat map. 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 heat map. A new dataframe is created comprising the chosen metadata.
47. Generate annotations for the heat map (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 (from Step 38) to assign colors to these unique groups.

```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")
```

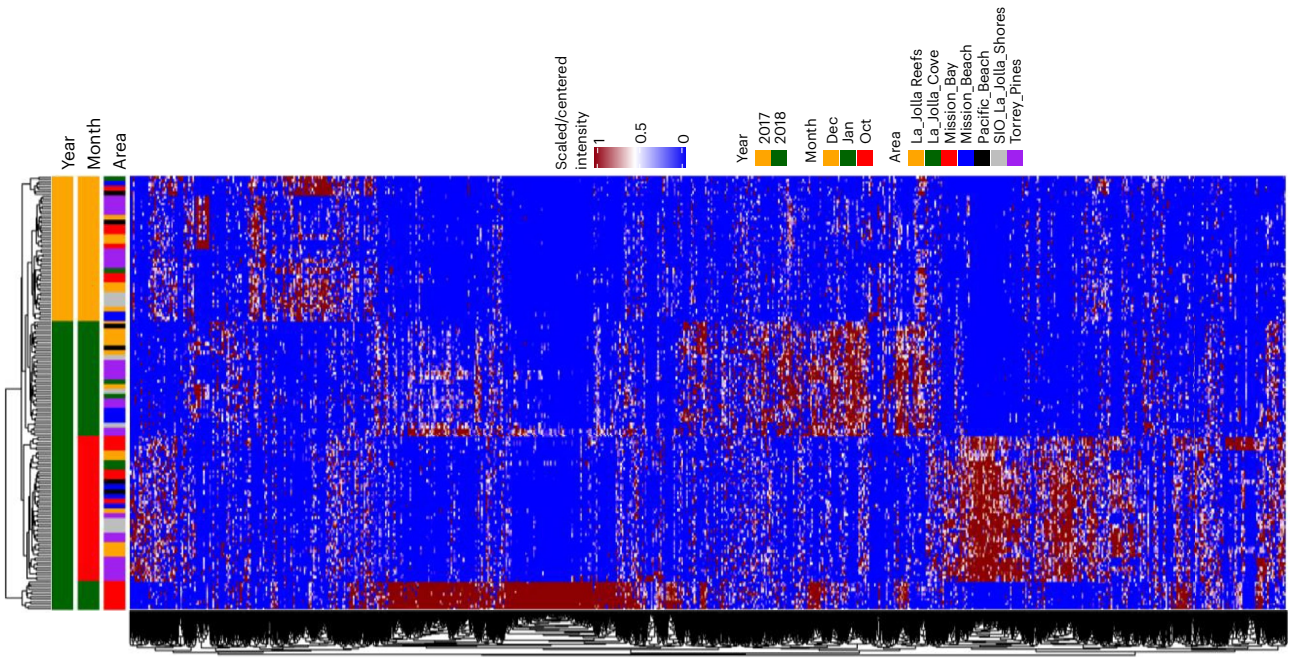


Fig. 10 | Heat map visualization and construction. The R code snippet used for heat map creation and the resultant heat map itself. To facilitate a comprehensive view, the heat map is oriented horizontally. The feature quantification table used here is the scaled table and feature intensities are color-coded, ranging from blue (0) to red (1). Annotations at the heat map's top delineate clustering based on variables such as year, month and sample area.

Users can modify these colors if desired in Step 38. After assigning colors to the subset dataframe, we use this information to decorate the heat map with annotations from the `HeatmapAnnotation()` function in the 'ComplexHeatmap' package.

48. Creating the heat map (user input—optional). To create the heat map, apply the 'heatmap' function from the ComplexHeatmap package on the transposed '`cleaned_data`' (as previously chosen in Step 30). This arranges the features in rows and samples in columns.

- For the heat map, 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 on the basis of specific needs
- The 'complete' linkage method is used for clustering (`clustering_method_rows = "complete",clustering_method_columns = "complete"`)

49. Refining data clustering with *k*-means. Further refine data clustering by incorporating the built-in *k*-means function within the heat map 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`).
50. Extracting features from each cluster. With a higher number of features, it is difficult to interpret the clustering or labeling of features on the heat map. To address this, we 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.

Supervised classification: RF

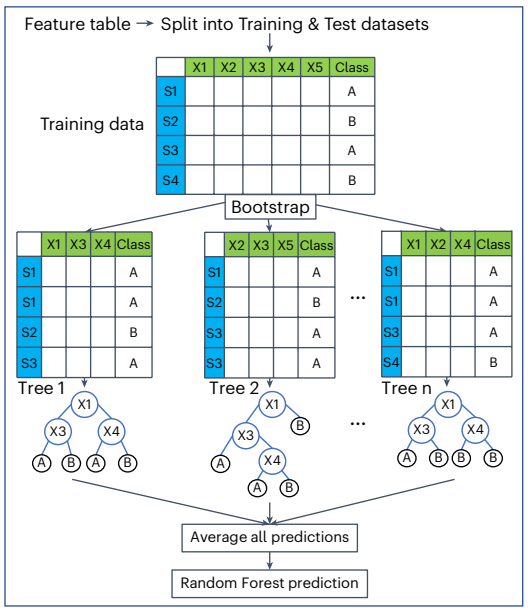
51. Prepare the data for RF. To do this:
    - First, load the '`rfpermute`' package
    - Start by merging the feature quantification table (in our example, '`Imp_s`' is chosen as the '`cleaned_data`' variable) and the corresponding metadata ('`md_Samples`') into a dataframe named '`cleaned_data_with_md`'. This step ensures that the samples are correctly aligned with their corresponding attributes in the metadata, which is essential for the subsequent analyses
52. Select the classification attribute for RF (user input required). Prepare the dataset used for RF classification so it only contains feature intensity information and the attribute of interest for classification. Here, we are classifying the samples according to different sample areas ('`ATTRIBUTE_Sample_Area`'); in this step, the user is prompted to input the index number of the interested attribute to use for the classification.
53. Balance sample sizes. If the sample size varies among the groups, balance the size of groups by using the `balancedSampsize()` function. This function helps ensure that groups with a larger number of samples do not disproportionately influence the model's outcome. It does this by selecting an equal number of samples from each group to include in each tree of the model, based on half the size of the smallest group[95]. For instance, in our example dataset, since the smallest sample count for a group is 12; thus, the function selects half this number (6) of samples from every group. This random sample selection is executed without replacement to maintain diversity in the model's training set. Samples not chosen for this process are designated as 'out-of-bag' (OOB) samples and are used for individual tree's model validation, ensuring a balanced representation, with at least half of the samples from each group being utilized for testing each tree's model's accuracy[124]. This ensures that each group is equally represented in the training process. The next step will delve deeper into how these balanced samples contribute to the model-building and validation process.
54. Run RF. This step introduces the execution of RF analysis using the `rfpermute()` function, designed to simplify the RF modeling process[124,125]. The function, an extension of the classic `randomForest()`, requires few parameters: the feature quantification table without the target classification column ('`x`'), the classification labels ('`y`'), a balanced sample size for each group ('`sampsize`'), alongside the specified number of trees (in our example, '`ntree`=500') and permutations (in our example, '`num.rep`=500'). Box 10 provides a detailed overview of RF along with a visualization of the RF algorithm and its

BOX 10

Random Forest (RF)

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. The prediction could be based on either class labels (classification) or numerical values (regression). RF algorithm enhances model generalization by training each tree on a different data sample, where the sampling is done with replacement (bootstrap sampling). Typically, each tree is trained on a bootstrap sample consisting of about two-thirds of the original dataset, while the remaining one-third, not included in the bootstrap sample, forms the OOB samples for that tree. These OOB samples act as a de facto test set for validating that tree's accuracy. This internal cross-validation method contributes to RF's

a Random forest




```
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

b

(continued from previous page)
robustness and supports its utility without necessitating a traditional test set[126,169]. These OOB samples provide an unbiased estimate of the model error as they were not seen by the tree during training. This unique utilization of OOB samples for error estimation and variable importance assessment through permutation makes RF particularly effective for a wide range of data analysis tasks[170].

However, the above-mentioned use of approximately two-thirds of the data for training each tree in an RF model is a tunable parameter, and users may adjust it to improve model performance. For example, in the '`randomForest`' function[127], the parameter that controls the bootstrap sample size is '`sampsize`'.

The OOB error rate is a measure of prediction accuracy and helps to improve the performance of weak or unstable learners in the model. While OOB error provides a good estimate of model performance, it may not entirely replace the need for a separate test set, particularly for evaluating generalization to new data. OOB estimates, although unbiased, may overestimate the model's error rate if not run long enough to reach convergence, in other words, it is crucial to train the RF with a sufficiently large number of trees until the error rates stabilize[126]. Acknowledging the diversity of practices in the field, with some using the traditional test-train split and others not[124,125], the model evaluation in RF can benefit from both OOB internal validation and testing the model on a separate test set for external generalization.

In RF, a common technique to assess the importance of each variable (or feature) in predicting the target classification is through permutation. 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[171].

Although permutation-based variable importance is a widely recognized method for evaluating the importance of variables in RF models, it is not a default method in all RF implementations. The 'rfPermute' package specifically facilitates this process for RF models in R, automating the calculation of importance scores by permuting feature values and assessing their impact on model accuracy. This approach helps in identifying the most influential variables and determining their statistical significance. To ensure consistency, we have implemented permutation-based variable importance calculations in both our R and Python notebooks.

In RF, there are two common metrics of variable importance used to rank features on the basis of their predictive power: 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 could be obtained by normalizing MDA, so they sum to 100, making them more interpretable on a relative scale[172].

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 versus predicted class labels), trace plot and check for overfitting by comparing testing versus training accuracy. However, supervised models may not be suitable for all datasets, 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.

RF algorithm visualization and execution. Panel **a** shows an illustration of the RF algorithm. The code block displayed here was used for model execution, using 500 trees and 500 permutations. Outputs of the rfpermute model, including a confusion matrix, are showcased. Panel **b** shows the OOB error curve, another crucial model evaluation metric.

implementation in R. With the `rfpermute()` function, a standard RF model is initially created to calculate the variable importance. Following this, the response variable (the metadata group '`ATTRIBUTE_Sample_Area`') undergoes permutation a specified number of times ('`num.rep`'). With each permutation, a new RF model is built to assess the impact of variable shuffling on the model's predictive accuracy[95]. `rfpermute()` streamlines the RF modeling process without necessitating a traditional train-test data split, such as the 70:30 or 80:20 ratio[126]. This approach takes advantage of OOB samples for individual tree's model validation by using data not selected during each tree's building phase. The `balancedSampsize()` function, discussed in the previous step, ensures equal representation from each group in the model to prevent bias. As a result, a portion of the data is utilized for training while the remainder serves as OOB samples for model validation. Once the model is run, the result displays the following:

- The number of variables tried at each split. This is the '`mtry`' parameter in the `randomForest` function. The value is 95 in our case. This is determined by the square root of the total feature count (9092), a default method in classification trees[127]

- The confusion matrix reveals an OOB correct classification rate (can be found under overall 'pct.correct') of ~68%. This represents the model's ability to correctly classify OOB samples, thereby providing an internal measure of performance
  ▲ **CRITICAL STEP** It is important to consider the following points when using the RF algorithm in R.
- The selection of the number of trees and permutations is crucial and should ideally be determined through hyperparameter tuning, such as with the '`randomForest`' package in R, to find the optimal settings. However, to keep the process straightforward for beginners, we use a value of 500 for both (see bullet below) and only the '`rfpermute`' package as a simplified approach. Also, '`rfpermute`' cannot be used for conducting traditional parameter tuning which is a limitation of this package
- 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 (for example, 500–1,000) and '`num.rep`' (500–10,000), then adjust on the basis of performance
- When working with large datasets, R may run out of internal memory trying to perform the RF. To work around this, adding the '`as.factor`' to the predictor variable (y), even if the class is already a factor, will alleviate the memory error

55. 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 model accuracy we refer to here is derived from an OOB estimate, providing an approximation of OOB sample accuracy, rather than the traditional comparison between training and testing set accuracies (that is, where the size of training and test data pools is a function of all available data instead of the result of balancing (see Step 53)). In our case, this OOB correct classification rate (overall 'pct.correct') was found to be ~68%
    - The confusion matrix is the most basic summary of a RF. See the accompanying image in Box 10 for reference. 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 changes as trees were added to the forest. See the OOB graph in Box 10 for reference. 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
56. Interpreting RF results. Beyond these, the RF results can be interpreted in various ways within the notebook. Users can generate the following results:
    - One could plot the most impactful predictors of the RF model using violin plots. Here, we show the top nine predictors in the notebook
    - Compare class predictions versus the actual group in a proximity plot. The group colors here are obtained from Step 38. This visualization is available in the notebook
    - Rank features by importance using the 'mean decrease accuracy' (MDA) 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' <0.05 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

Univariate statistics

● **TIMING 50–60 min**

Start by installing the packages necessary for this section: FSA[96] (v0.9.4), matrixStats[97] (v0.63.0).

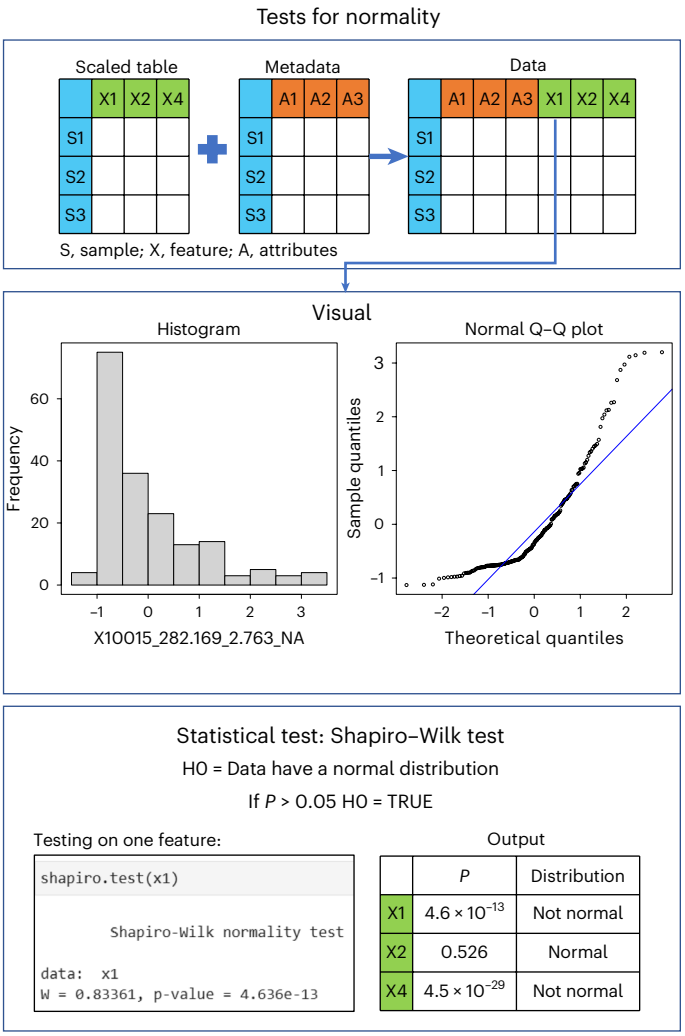


Fig. 11 | Assessing normality of features. Methods to assess normality for individual features. It showcases visual approaches like histograms and Q–Q plots, where deviations from normality can be visually assessed. The third segment delves into significance testing using the Shapiro–Wilk test, emphasizing that a *P* value greater than 0.05 suggests a normal distribution.

Test for normality

▲ **CRITICAL** 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. A graphical representation of testing normality of features is shown in Fig. 11. To know more about normality assumptions, refer to Box 11. Figure 12 provides a flowchart that guides the selection of appropriate statistical tests based on data normality and homogeneity.

57. 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. The feature for this example is chosen from the '`cleaned_data_with_md`' dataframe, which was prepared in Step 51. Step 58 also demonstrates how log-transforming the data can improve the normality of the data.
58. Normality testing for all features. Perform a Shapiro-Wilk test for each feature and record the resulting *P* values. Correct these *P* values for FDR using the BH 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 nonparametric tests for further analysis. In our example data, out of the 9,092 features, only 54 had a normal distribution. Thus, performing nonparametric tests, such as the KW test, might make more sense for our data.

BOX 11

Normality assumptions

Besides normality, it is essential to consider two other critical assumptions when deciding between parametric and nonparametric 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 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[129,173].

Parametric and nonparametric tests

▲ **CRITICAL** In this procedure, we performed both ANOVA (parametric) and KW tests (nonparametric), along with their respective post hoc tests on the same dataset. The only reason that both tests were performed was to demonstrate to users how these tests can be applied and what the potential results might look like. However, only one of the two tests is necessary depending on whether the user's data conforms to parametric test assumptions. Therefore, it is advisable to choose the test that best suits your data's characteristics rather than employing both. The KW test is considered more appropriate for the example dataset due to the non-normal distribution of the majority of its features (Step 59). For more information on which test to choose on the basis of normality assumptions, refer to Fig. 12.

Parametric tests

▲ **CRITICAL** For all the tests below, 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.

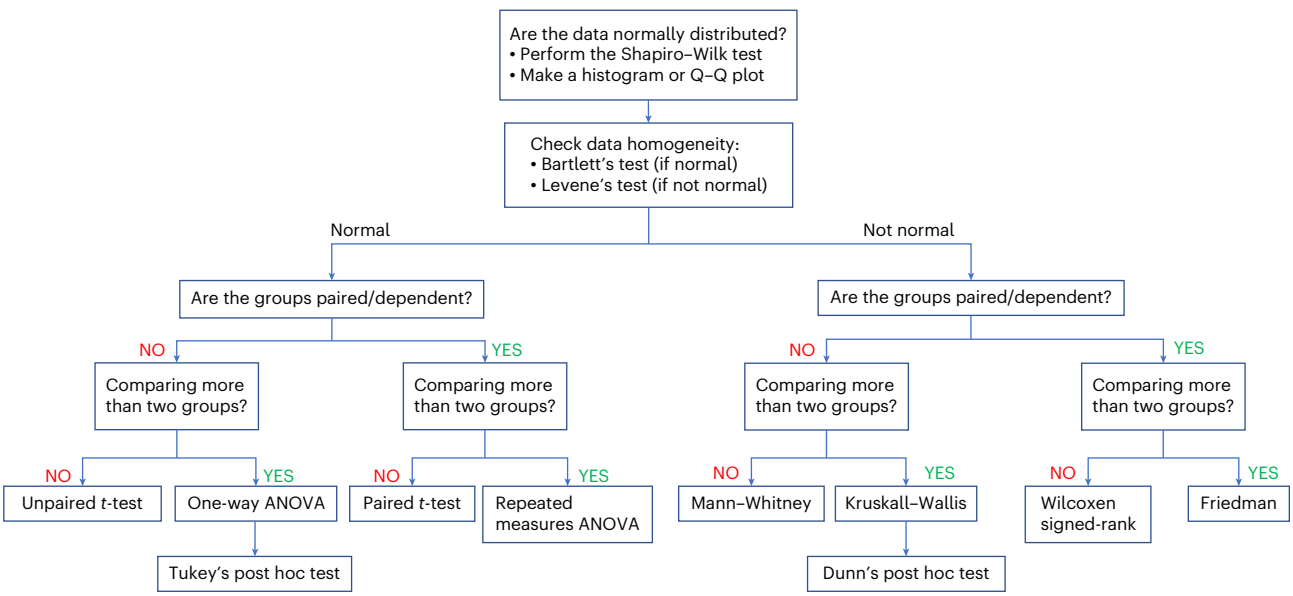


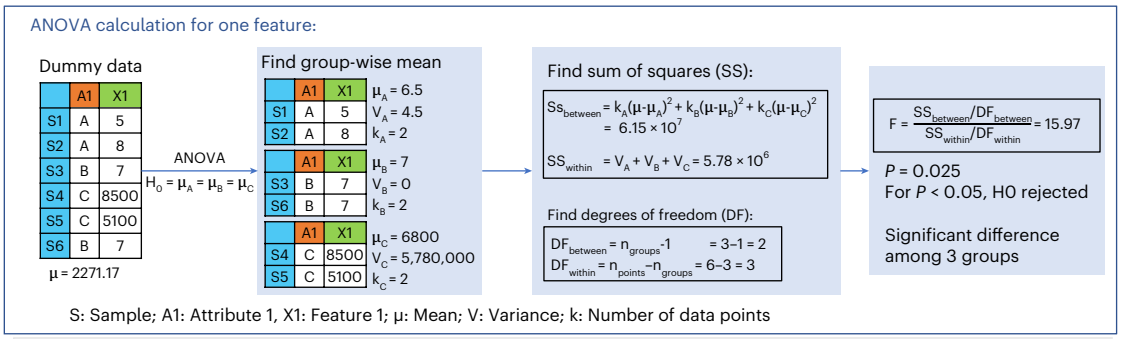
Fig. 12 | Selection of statistical tests for univariate analysis. The flowchart guides users in choosing the appropriate univariate statistical test, starting with a normality test (for example, Shapiro–Wilk). If the data are not normally distributed ($P \leq 0.05$), nonparametric tests are recommended. Homoscedasticity, or equality of variances, can be checked using Levene's or Bartlett's tests. The flowchart then directs the choice between parametric and nonparametric tests, based on whether the data groups are paired or unpaired and on the number of groups being compared.

BOX 12

ANOVA

If a pairwise test is used (for example, 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[174]. 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 (for example, origin, location, type of soil and so on). The *F*-statistic is calculated using the sum of squares and the degrees of freedom (seethe illustration below) and compared with a standard *F*-distribution to determine whether the differences among group means are greater than

a



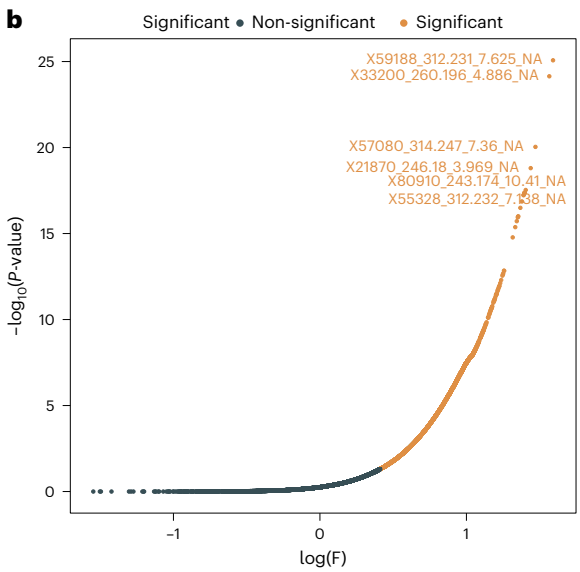

```
anova_group <- uni_metadata[,interested_attribute_anova]
anova_group <- as.factor(anova_group) # convert the attribute to 'factor' type

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

b



(continued from previous page)
would be expected by chance. Importantly, the alternative hypothesis (that is, where a difference exists between the means) is unspecific. This means that the test does not tell us where the difference(s) lie (for example, 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[129,175]. Nonparametric tests should be used if these assumptions do not hold in the data of interest.

The illustration depicts the ANOVA process applied to a sample feature across groups A, B and C using dummy data. Following this, the R code block used to test ANOVA on our dataset is displayed. This process involves selecting metadata for grouped information (for example, 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 the negative logarithm of *P* values.

ANOVA test

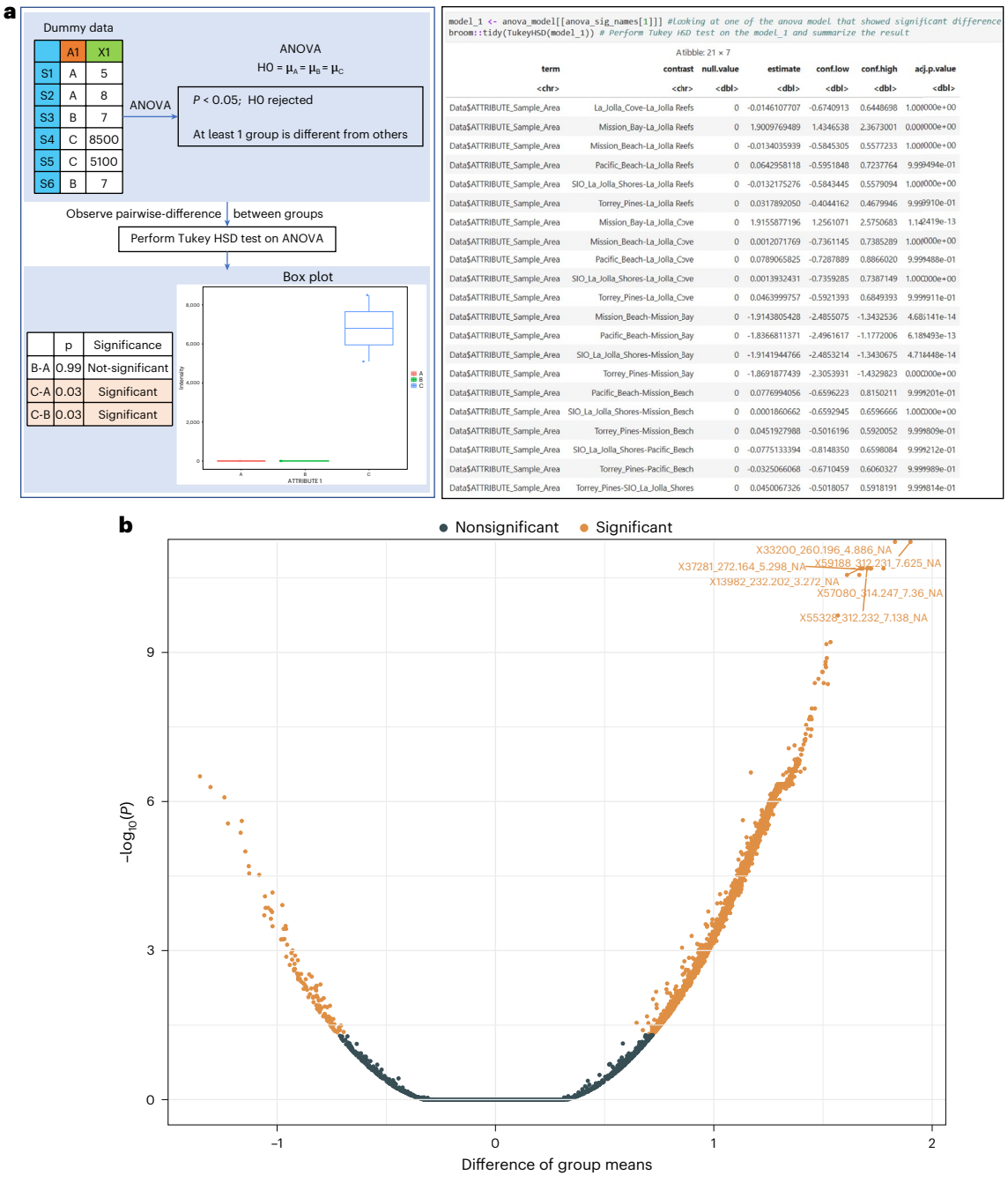
▲ **CRITICAL** The ANOVA is the statistical procedure used to test if there exists a significant difference in the means of a dependent variable between three or more groups. As opposed to a pair-wise comparison where we compare the means in a variable (that is, $\mu_1 = \mu_2$), in the ANOVA we compare the means of several groups[67]. For a deeper understanding of ANOVA, please refer to Box 12. Furthermore, the accompanying illustration 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.

59. Run 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.
60. Run 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[128] 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.
61. Subsetting significant features. Filter out the significant features for further examination. Display the count of significant and nonsignificant features.
62. 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 the 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 six significant features.
63. Visualize the top significant metabolites. Generate boxplots for the top four significant metabolites to observe how their intensity levels differ across sampling sites. The colors for these different sampling sites are generated from Step 38. 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 13

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[129]. The most used post hoc test for ANOVA is Tukey's 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[176].



(continued from previous page)

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 this, we present the code block demonstrating the Tukey test applied to the first significant feature identified via ANOVA. Given the presence of seven 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'.

Tukey's HSD test

▲ **CRITICAL** 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(s) exist. To do this, we can conduct a Tukey's honestly significant difference (HSD) post hoc test to compare multiple means in a single analysis[129]. Refer to Box 13 for more information on Tukey's test. Additionally, the box 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.

64. Perform Tukey's HSD for a significant feature. First, we select a feature identified as significant in the ANOVA result, using '`anova_sig_names`' generated in Step 62. From the ANOVA output, we subset the data for this significant feature and conduct a Tukey's HSD test. The output is a comprehensive table providing an assessment of every possible pairwise group difference as shown in the code snippet in Box 13. To conduct a Tukey's 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.
65. Perform Tukey's HSD for all significant features (user input required). Carry out a Tukey's HSD test for all the significant features identified in the ANOVA. Then, filter the results for the specific comparison such as 'Mission Bay versus 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's 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.
66. Count significant features. Determine how many features exhibit a significant difference between the chosen sites and how many do not.
67. 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 whether the significant metabolite is upregulated or downregulated among the chosen sites. The colors for the different groups in these boxplots are obtained from Step 38.

***t*-tests**

68. 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 two months, Dec 2017 and Oct 2018 (without rainfall).
    ▲ **CRITICAL STEP** This '`ATTRIBUTE_rainfall`' column addition caters to our dataset's context. This example aims to illustrate the concept of including samples' environmental context rather than serve as a model for this test. Users with preexisting binary attributes can skip this addition, while others may adjust this step to align with their data.

69. Perform *t*-test. Following the same steps as ANOVA (from Steps 60–63), 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.
70. 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 two 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. The colors for the different groups in these boxplots are obtained from Step 38.
    ▲ **CRITICAL STEP** 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.

Nonparametric tests

▲ **CRITICAL** Nonparametric tests can be performed when parametric tests are not appropriate due to data characteristics, such as non-normality. For all of these 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.

KW test

▲ **CRITICAL** The KW test is a nonparametric 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[130]. For more information on the KW Test, refer to Box 14. Additionally, the accompanying figure shows a visual explanation of the KW algorithm, along with the R-code used to test a feature across various groups and determine its significance.

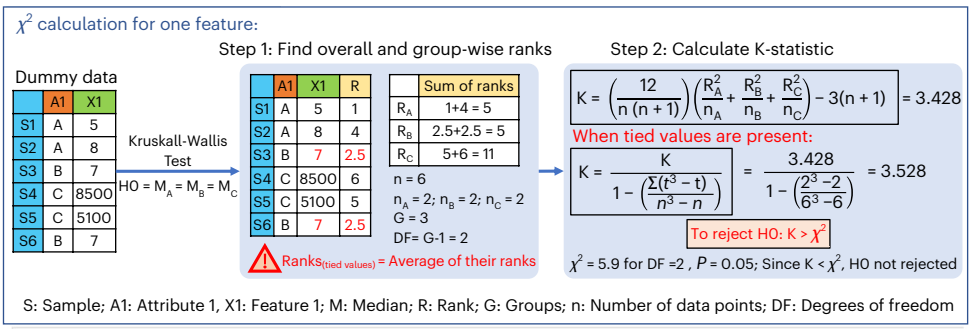
71. Perform KW test on one feature (user input required). Begin by specifying the attribute for the KW test by entering its index number. In this tutorial, we opt for '`ATTRIBUTE_Sample_Area`'. Then, apply the KW test on a single feature (the first feature in the '`uni_data`' dataframe) across different 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 (Step 30). Summarize the output into a one-row table using the `tidy()` function from the broom[128] package as shown in the figure. The steps for the KW test (Steps 72–75) are structured similarly to the ANOVA steps (Steps 60–63).
72. Run KW test for all features:
    - Just as in ANOVA (Step 61), perform the KW 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
73. Filter significant features. Display the count of significant and nonsignificant features. Filter out the names of significant features from the KW output dataframe for further analysis. This is done by selecting the column 'significant' in '`kruskall_out`' dataframe and filtering rows labeled 'significance'. Extract their row names, which include information such as unique feature ID, *m*/*z* values, RT and annotation if available.
74. Visualize KW results. Similarly to visualizing 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 six significant features.
75. Visualize the top significant metabolites of KW results. Generate boxplots for the top four significant metabolites to observe how their intensity levels differ across sampling sites. The colors for the different sampling sites in these boxplots are obtained from Step 38. 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.

BOX 14

Kruskal–Wallis (KW) test

Although the KW 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[130]. 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[177]. With this information, the *K* statistic can be calculated and compared with the chi-square distribution to accept or reject the null hypothesis (see the illustration below). If the null hypothesis is rejected, the alternative hypothesis states that at least one group has a different median from the others.

The illustration provides a comprehensive view of the KW 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 KW test is executed on an individual feature in relation to the metadata column that groups information, with our primary interest being the 'sample area'.




```
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

Dunn's post hoc test

▲ **CRITICAL STEP** The Dunn statistical test is a nonparametric alternative to the Tukey's HSD post hoc test to make pairwise comparisons between multiple groups. The steps for Dunn's post hoc test (Steps 77 to 80) are structured similarly to the Tukey's HSD steps (Steps 65–68). Refer to Box 15 for more information on Dunn's post hoc test. The accompanying figure in Box 15 shows a visual representation of applying the Dunn test and its implementation in R.

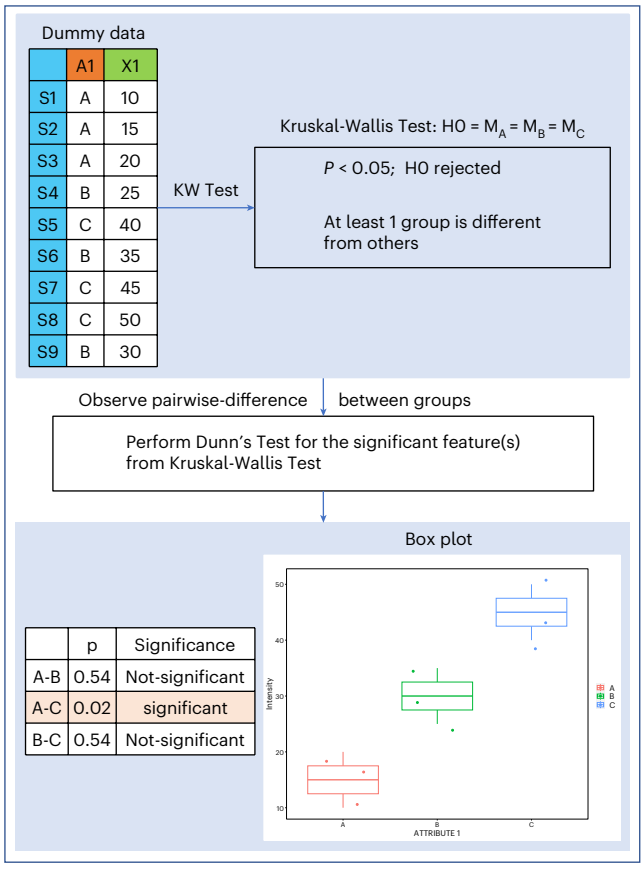
76. Perform Dunn test for a significant feature. First, we select the first feature identified as significant in the KW test result, using '`kw_sig_names`' generated in Step 74. From the KW output, we subset the data for this significant feature and conduct a Dunn test using `dunnTest()` function. The output is a comprehensive table providing an assessment of every possible pairwise group difference as shown in Box 15. When conducting a Dunn test on all significant features, consider specifying just one pair interaction to maintain simplicity. Similarly to the Tukey HSD test section, here we will focus on the results from comparisons between 'Mission Bay' and 'La Jolla Reefs' in the subsequent step.
77. Perform Dunn test for all significant features (user input required). Carry out a Dunn test for all the significant features identified in the KW test with BH correction for *P* values. Then, filter the results for the specific interaction 'Mission Bay vs. La Jolla Reefs'. To perform

BOX 15

Dunn test

The Dunn statistical test is a non-parametric post-hoc test following KW test similar to the Tukey's 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.[178]

The image illustrates the Dunn test, a post hoc analysis following the KW test. After identifying significant features from the KW 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 KW test, examining its relationship with various sample areas. The resulting display includes *P* values for all potential 21 pairwise comparisons.



	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	2.60400633	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

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 on the basis of the corrected *P* values (`dunn_output$P.adj < 0.05`) to identify the features that show a significant difference between these two sites.

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

79. 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 two 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. The colors for the different sampling sites in these boxplots are obtained from Step 38.

Integrating statistical results into a molecular network

● **TIMING 1–2 h**

▲ **CRITICAL** Statistical outcomes and generated CSV files can be integrated into a molecular network via Cytoscape to help prioritize subsequent MS annotations and full structural identifications. For instance, in this protocol, we demonstrate the integration of RF analysis results with the GraphML file obtained from FBMN, which is accessible in Cytoscape. The file used for this section is the 'Significant_features_RF' file generated in Step 56, and it contains significant features with a 'MeanDecreaseAccuracy.pval' less than 0.05. The procedure for generating and interpreting this ranked list of significant features is described in Step 56. The file can be accessed from our GitHub repository (https://github.com/Functional-Metabolomics-Lab/FBMN-STATS/blob/main/R/outputs_R_Notebook/Multivariate_results/2023-09-07_Top5percent_ImportantFeatures_RF_500trees_500perm.csv).

Preparing the CSV File for Integration

80. Start by adjusting the first column of your CSV, which typically contains concatenated identifiers (for example, X91133_907.259_12.628_NA;THEAFLAVIN DIGALLATE) into a single string. This column represents a unique ID, m/z, retention time, and annotated names.
81. Use the delimiter '_' to separate these components into four columns and rename the columns to 'uniqueID', 'mz', 'RT' and 'annotated_names' for clarity. Retaining these unique identifiers as the first column in the CSV facilitates their use as a key for integration with the Cytoscape GraphML file.
82. Remove the 'X' prefix on the first column 'uniqueID'
83. Add a column labeled 'RF significant features' and fill it with 'RF significant features'. This can be used later in Cytoscape to mark the RF-identified significant features.
84. Once modifications are complete, save this modified CSV for subsequent integration with Cytoscape. Reference for the modified file is available at: https://github.com/Functional-Metabolomics-Lab/FBMN-STATS/tree/main/Integrating_Stats_Results_to_Molecular_Network.

Integrating the data into Cytoscape

85. After your FBMN job is complete, download the Cytoscape data from the GNPS status page under 'Export/Download Network Files' and click 'Download Cytoscape Data'. This zip file contains all the necessary files for integration.
86. Launch Cytoscape and open the GraphML file from your GNPS download.
87. Import the modified CSV: use the 'Import Table from File' feature in Cytoscape to bring in your modified CSV file (Fig. 13a; icon of a table and downward pointing arrow). In the 'Import Columns from Table' dialog, ensure the settings match those depicted in Fig. 13b:
    - Import data as: Node Table Columns
    - Key column for network: choose 'shared name' to align with the 'shared name' column in the original node table
    - In the 'Preview' section, make sure the first column of the imported table, 'uniqueID', is assigned as the key column to merge
    - Verify the type of both these columns to confirm that they are both numeric. Different types will disturb merging

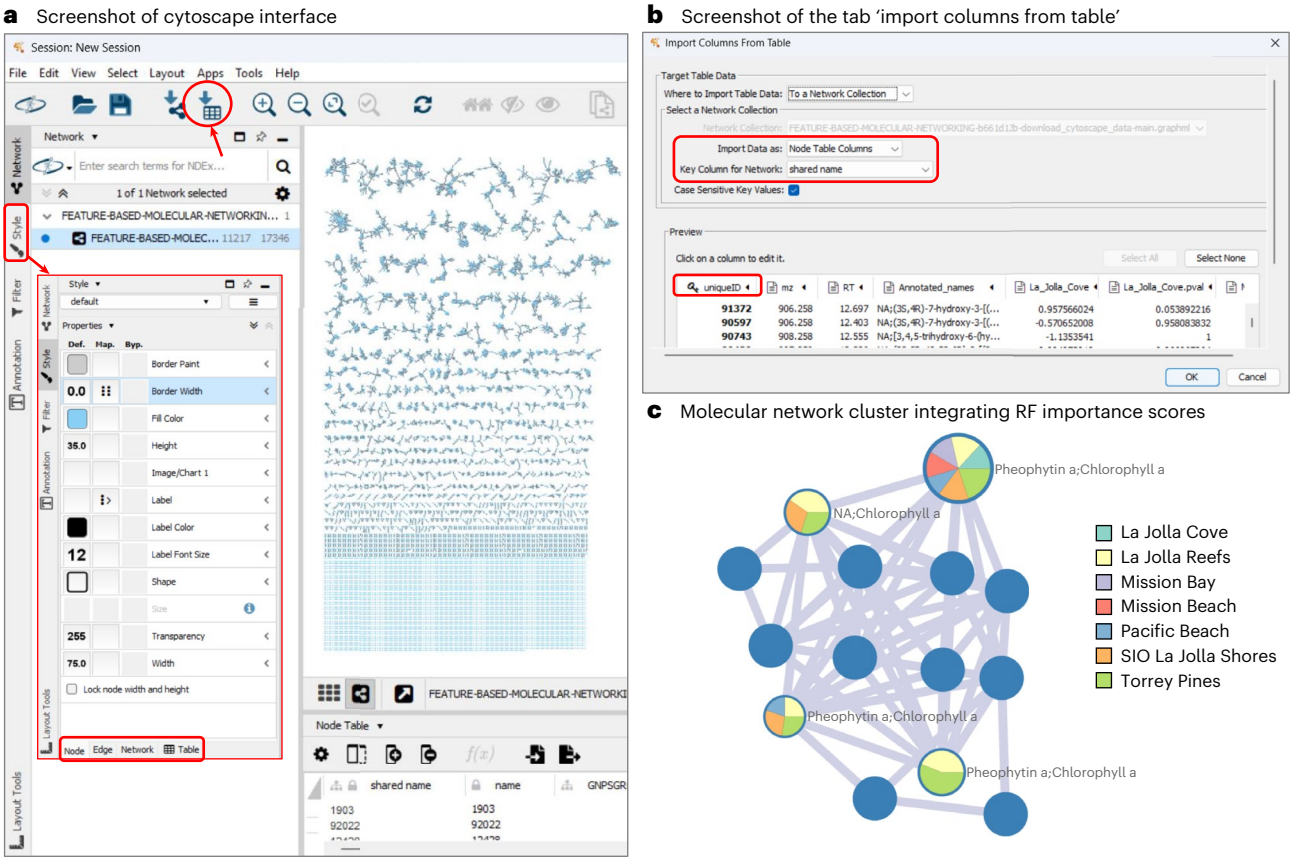


Fig. 13 | Visual guide to integrating data into Cytoscape networks. **a**, Screenshot of the Cytoscape interface showcasing the initial view upon loading a GraphML file from FBMN results. Key areas are outlined in red. On the left sidebar with primary icons (Network, Style, Filter and Annotation), 'network' is currently active (emphasized in gray) and the 'style' icon is expanded to reveal options for customizing nodes, edges and more. In addition, the icon 'Import Table from File' from the top menu bar is also circled. **b**, After selecting 'import table from file', the user navigates through a dialog box ('import columns from table') that allows for the merging of the modified csv table into the existing network based on a unique index column present in both tables. Crucial settings such as 'import data as: node table columns' and 'key column for network: shared name' are outlined in red. Below, the 'preview' section shows the modified RF results table, with 'uniqueID' (column containing unique feature IDs) as the key column for merging. **c**, The figure displays a small example network where nodes are enhanced with RF importance scores. Nodes without statistical significance are depicted in blue, while significant nodes showcase a pie chart visualization. Each segment of the pie chart represents the importance score for one of the seven locations (as an example, the seven locations share similar importance scores for the uppermost significant node, while Torrey Pines has the highest importance score for the lowermost significant node). The overall node size corresponds to the MDA metric.

Visualizing RF results

88. To represent MDA scores visually, adjust node border widths under the 'Style' tab in Cytoscape (Fig. 13a): Navigate to Node → Border Width and select your MDA column for mapping, choosing a passthrough mapping type.
89. Consider differentiating significant RF features, such as changing their shape to squares, by utilizing the 'RF significant features' column. This can be done by navigating to Style → Node → Shape, selecting the 'RF significant features' column under 'column' and setting 'mapping type' to 'discrete mapping'. Assign shapes, such as rounded squares, for significant features, as shown in Fig. 14g. This is particularly useful if you are incorporating results from several statistical tests.
90. For a comprehensive view, pie charts can illustrate the importance score of each feature across groups (for example, seven locations) such as in Fig. 13c. This visualization can highlight features with varying significance across the groups, aiding in the prioritization

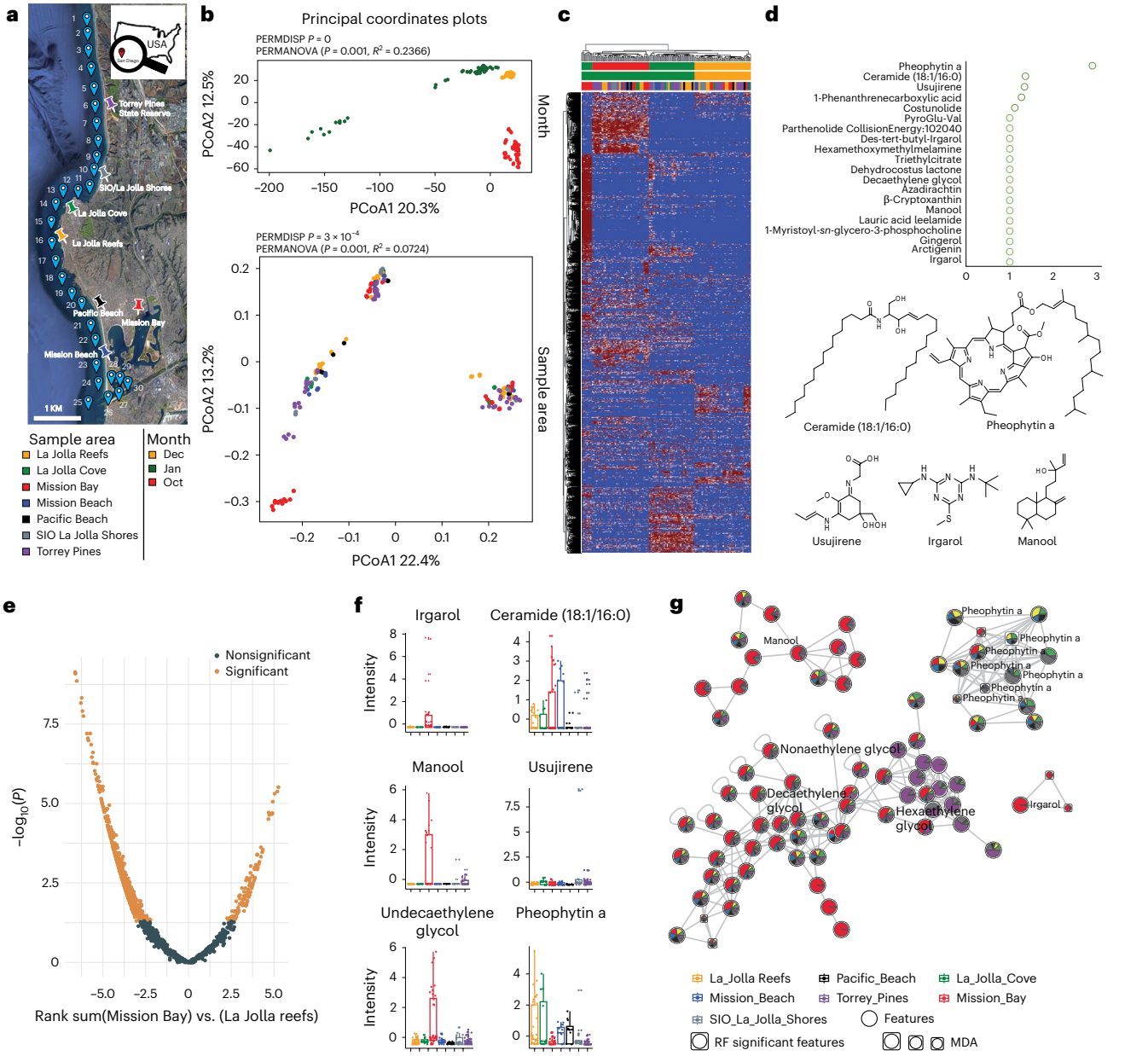


Fig. 14 | Anticipated results. a, Spatial map pinpointing sampling sites. **b**, Principal coordinate plots delineating differences by sampling month and location. **c**, Heat map displaying scaled feature intensities. **d**, Top 20 annotated drivers for temporal changes identified via RF, with structures of the top three metabolites shown. **e**, Volcano plot illustrating the Dunn test comparison between Mission Bay and La Jolla Reefs samples, with features deemed significant in the KW sample area-based test used for this post hoc analysis. **f**, Box plots illustrating feature intensities across various sampling locations. The first column presents the foremost three annotated significant outcomes post-Dunn test, accompanied by their molecular structures. The second column highlights the top three significant outputs as identified by RF. **g**, Example molecular networks of statistically significant features identified by the KW test (for example, Irgarol and Manool) and RF analysis (for example, pheophytin a and decaethylene glycol). Each node in the network represents a feature and is visualized with a pie chart representing its intensity distribution across seven sample locations. Nodes bordered in a square are deemed significant by the RF model. Their node size corresponds to MDA scores from the RF analysis, where a larger size denotes a higher significance score.

process. To create a pie chart, click on the pie chart icon in the Node section's style menu, labeled 'image/chart 1'. A window will appear where all the available columns can be selected (we used the RF importance scores for each location) and a 'customize' button, where each feature can be assigned a color.

Troubleshooting

Troubleshooting advice can be found in Table 4.

Table 4 | Troubleshooting table

Step	Problem	Possible reason	Possible solution
2, 26, 31, 51, 57	Package installation fails or is slow	'p_load' function from pacman library may not retrieve some packages well	In such cases, install and load the packages manually: `install.packages('tidyverse')` `library('tidyverse')`
3	Incorrect path entry in code cell and it results in error	Users enter the folder path directly in the code cell	Run the code cell first (shift + enter in case of the JupyterLab environment or simply press the play button next to the code cell in terms of Colab environment). This will result in a pop-up box asking for path entry. Here, enter the path of the folder that contains the input files. Make sure to run this step and set a working directory as all the resulting files will be stored here
3, 4	While setting the working directory in Google Colab, users enter a local path and it fails	Google Colab platform cannot access the local files directly. One can mount their Google Drive and access files from there, but that is possible only with Python and not in R	Make sure to create a folder in the Colab platform and upload your input files there manually, then set the path of that folder as the working directory
6	Metadata table not read correctly	Metadata was uploaded in incorrect format	The R notebook expects the metadata to be in txt or tsv format. So make sure to save it that way and not in csv mode before uploading the file
			Else, in Step 5, you can change the separator `sep = '\t'` in the following code block to `sep = ','` or `';'`: `md <- read.csv(file_names[input[2]], header = T, check.names = F, sep = '\t')'`
6	Files not read or separated correctly	Incorrect file formats and separators	Ensure the feature table is '.csv' and others are '.tsv' or '.txt'. Check that '.csv' is comma-separated and not semi-colon separated because of your regional language settings
17	Filenames in the metadata are missing in the feature table and this step lists the missing filenames	Spelling errors, case sensitivity issues, and the presence of whitespaces in metadata filenames are common issues, particularly as metadata is often manually entered in Excel	Check the filenames in metadata against the corresponding column names in the feature table for spelling mistakes and case errors. Then reupload the file
30	Error suggesting missing dataframe	Missing dataframes from the previous clean-up steps	Whether the user wants to perform the data clean-up steps or not, execute all preceding steps to make sure necessary dataframes are available to be retrieved for this step. Here, the user can choose which of these dataframes to be used for the statistical analysis
35	PCoA plot fails due to color limitations	In this example, the column used for coloring scores, 'ATTRIBUTE_Month', is preset based on the example dataset. Errors may occur if this column name is not updated when using different data. Additionally, plotting issues may arise if the chosen metadata column has more than the eight available colors, leading to coloring errors in the scores	Make sure to choose a metadata column with eight or fewer groups, or customize the coloring in the code to fit the number of groups
39	Color error in plots	Here, the colors are specified for five groups. If the chosen attribute has more than five groups, it might cause an error	Specify colors for all groups or allow the function to use default settings by removing the cols command. The plotting function plotPCoA can handle up to 22 groups
48, 49	Heat map visualization issues with non-scaled data	`t(cleaned_data),` `heatmap_legend_param = list(title = "Scaled/centered\nintensity"),` `col = circlize::colorRamp2(c(0, 0.5, 1)),` The above code within the 'heatmap' function states that we are using the transposed cleaned data (which is scaled data, in the example) and the color ramp is given as 0, 0.5,1 When using non-scaled data for a heat map, the specified color ramp may not adequately represent all intensity variations, resulting in poor visualization quality	If you chose a different dataframes (such as raw data, or just blank removed data or TIC normalized data) as your cleaned data at Step 30, then adjust color ramp settings accordingly in these steps as `circlize::colorRamp2 (c(min(cleaned_data), 0, max(cleaned_data)))` Consider using a color (for example, yellow) for NA values if present by including the following within the function: `'na_col = 'yellow''`

Common troubleshooting tips for the R notebook can be found in Table 4 and within each step of the main protocol as needed. Additionally, the Supplementary Information document provides insights on differences between the R steps and other notebooks, as well as the fbmn-stats app, along with further troubleshooting advice. For any unresolved issues, we recommend submitting an issue on our GitHub page.

Timing

The processing times reported here are based on using our example dataset within the Colab platform, with durations estimated from a beginner's perspective. This reflects the time typically required for someone new to complete the analysis. Running these procedures on local Jupyter Notebook systems could substantially decrease these times.

The preliminary setup for the notebook, covering Steps 1–17, takes ~30–40 min, with the package installation alone taking ~10–15 min. The data cleaning process, spanning Steps 18–30, is estimated to take 20–30 min. Multivariate statistical analysis, which includes Steps 31–56, generally requires 50–60 min, with heat map generation and RF classification taking the longest time—5–10 min for each heat map generation and 30–60 min for RF.

Univariate statistics, covering Steps 57–80, also take ~50–60 min, with specific Steps 59, 61, 66, 73 and 78 each taking 5–10 min. Finally, integrating statistical results into a molecular network, detailed in Steps 81–91, could take 1–2 h for users who are new to Cytoscape, as they learn to navigate and utilize the software effectively.

Anticipated results

We illustrate the types of results that could be obtained using specific worked example.

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, California, during different dry and wet seasons. Refer to Fig. 14a for a spatial map of the sampling locations. The presumption was that postrain samples from January 2018, influenced by runoff, would show increased pollutant levels. From FBMN analysis, we identified 5,521 LC–MS/MS features, which decreased to 4,384 after removing blanks. The library search against the GNPS spectral library via the FBMN workflow resulted in 92 annotated features out of the 4,384 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 (December 2017, January 2018, October 2018) and two PPL process blanks for each sample time. From this extended dataset, we identified 11,217 features, with 260 GNPS library matches and 1991 analog matches. When focusing solely on December and January samples, the feature count surged to 10,470, almost double the original count of 5,521 features, 240 GNPS library hits and 1,624 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 (version 3.3.0) and extended our SIRIUS analysis using tools like CANOPUS and CSI:FingerID. The SIRIUS result provided annotations for 8,255 features, with annotations or compound names available for 5,001 features. All 5,001 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 8,885 features, with 5,411 of these having an explained intensity greater than

80%, marking them as reliable formulas. For compound class predictions, CANOPUS provided annotations for 8,583 features spanning various levels, such as kingdom, superclass, class, subclass and level 5. On the other hand, the Natural Product Classifier 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 subsetting of features for subsequent statistics.

Impact of sequential data clean-up

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 Fig. 14b. Postblank removal intensified these groupings. Before data clean-up, 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 $R^2 = 0.34$). Sample locations, however, explained only 7% of the variance. Upon deeper exploration of the metabolic profiles across these sampling locations, January's variance was more prominent in Mission Bay, especially postrainfall, due to its nutrient-rich status, potentially from increased runoff through the San Diego River. This distinction is evident in the PCoA plot in Fig. 14b. Our data showed Mission Bay's pre-rainfall samples were similar to those of other sites, but postrain samples in January diverged—a pattern absent in December 2017 and October 2018 samples. We could also observe some clear patterns in the heat map depicted in Fig. 14c. 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 with 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 RF classification considering sampling sites.

RF exploration: prioritizing key drivers

In our example provided in the notebook, we tried to classify surface seawater samples based on their different sampling sites using RF. Here, the feature quantification table without metadata is the predictor variable, and the metadata group 'sampling site' is the response variable. The figure in Box 10 provides a visualization of the RF algorithm and its implementation in R. Utilizing a RF 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 5 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 the 'MDA *P* value'). Of these, 7 matched GNPS libraries, and 96 were analog hits. Examining the violin plot results of RF, top features, like those with library identifications 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 digallate (ID 91133). Some features, such as IDs 33200 and 53617, were notably elevated in Mission

Table 5 | Confusion matrix of RF classification

	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

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.

Bay alone. Certain compounds from previously reported research, such as m/z 1129.3145 (analog name: benzyltetradecyldimethylammonium) 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 14d visualizes the top 20 annotated drivers for site-specific changes as identified via RF, highlighting the structures of the top 5 metabolites. In summary, our extended dataset enhances the RF analysis, offering a detailed understanding of chemotype differences across coastal areas and reaffirming the conclusions of the original study.

Univariate analysis insights

In our univariate analysis of 9,092 features, we used both ANOVA and the KW tests to demonstrate to users how these methods can be utilized with the data and to offer insight into the potential results. However, the KW test was considered more appropriate due to the non-normal distribution of most features' relative intensity values in our dataset. The KW test highlighted 1,258 significant features, including irgarol, an antifouling agent used on boats. Conversely, ANOVA pinpointed 1,554 significant features, with many features having a pronounced abundance in Mission Bay compared with other sites.

Building on the KW results, Dunn's post hoc 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 nonsignificant features from this test are visualized in the volcano plot in Fig. 14e. Notably, compounds like irgarol and manool were significantly higher in Mission Bay. In contrast, La Jolla Reefs had a higher presence of the natural product pheophytin a. The top row in Fig. 14f displays the intensities of the top three annotated results from the Dunn test across the sampling locations using box plots. These findings align with and reinforce the initial observations, validating the robustness of our analytical workflow.

Integration of MN 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 14g shows the networks of 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. It is important to note that these are example networks selected from the larger overarching FBMN network. The resulting Cytoscape files for these and the complete network are available on our GitHub repository (https://github.com/Functional-Metabolomics-Lab/FBMN-STATS/tree/main/Integrating_Stats_Results_to_Molecular_Network)

for further reference. These results show nicely how statistical prioritization and further structure-based similarity (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 from the supervised multivariate analysis can be imported as a new attribute to the networks with tools such as Cytoscape to combine visual and statistical prioritization directly in the network. For guidance on this integration process, please refer to the 'Integrating statistical results into a molecular network' section in Procedure.

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 scientific 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, such as the number of trees in a RF 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 compounds is just the beginning. To fully understand the relationships between metabolites, xenobiotics 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 related biological processes. When specific compounds are of particular interest, targeted metabolomics stands as a powerful next step.

The versatility of our protocol extends to a wide range of fields and sample types, including combinatorial chemistry, doping analysis, and trace contamination of food, pharmaceuticals, and other industrial products. It is equally applicable to biological samples from diverse origins, such as microbiomes, bioreactors or biomedical research, provided the data adheres to the feature table and metadata format specifications. Beyond sample classification and feature prioritization, our protocol facilitates the integration of statistical findings into molecular networks, allowing users to visualize complex chemical spaces across various research domains.

In summary, we anticipate that our 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.

Reporting summary

Further information on research design is available in the Nature Portfolio Reporting Summary linked to this article.

Data availability

The FBMN results are available at https://gnps.ucsd.edu/ProteoSAFe/status.jsp?task=b661d12ba88745639664988329c1363e. Raw and processed data are available through the MassIVE repository, MSV000082312 and MSV000085786, and through Zenodo (https://doi.org/10.5281/zenodo.10051610).

Code availability

All code and software is available through GitHub (https://github.com/Functional-Metabolomics-Lab/FBMN-STATS). The web application can be accessed at https://fbmn-statsguide.gnps2.org/. Downloadable Windows executables of the web app is available from https://www.functional-metabolomics.com/resources. All the code is deposited on Zenodo at https://doi.org/10.5281/zenodo.11350947.

Received: 31 October 2023; Accepted: 2 July 2024;
Published online: 20 September 2024

References

1. Vailati-Riboni, M., Palombo, V. & Loor, J. J. What are omics sciences? in *Periparturient Diseases of Dairy Cows* (ed. Ametaj, B.) Ch. 1 (Springer, 2017); https://doi.org/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); https://doi.org/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. Usa.* **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**, 109449 (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. USA* **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. Kessner, D., Chambers, M., Burke, R., Agus, D. & Mallick, P. ProteoWizard: open source software for rapid proteomics tools development. *Bioinformatics* **24**, 2534–2536 (2008).
41. Hulstaert, N. et al. ThermoRawFileParser: modular, scalable, and cross-platform RAW file conversion. *J. Proteome Res.* **19**, 537–542 (2020).
42. Adusumilli, R. & Mallick, P. Data conversion with ProteoWizard msConvert. *Methods Mol. Biol.* **1550**, 339–368 (2017).
43. 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).
44. 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).
45. Schmid, R. et al. Integrative analysis of multimodal mass spectrometry data in MZmine 3. *Nat. Biotechnol.* **41**, 447–449 (2023).
46. Tsugawa, H. et al. A lipidome atlas in MS-DIAL 4. *Nat. Biotechnol.* **38**, 1159–1163 (2020).
47. Pfeuffer, J. et al. OpenMS—a platform for reproducible analysis of mass spectrometry data. *J. Biotechnol.* **261**, 142–148 (2017).
48. Gloaguen, Y., Kirwan, J. A. & Beule, D. Deep learning-assisted peak curation for large-scale LC–MS metabolomics. *Anal. Chem.* **94**, 4930–4937 (2022).

49. 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).
50. 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).
51. Heuckeroth, S., Damiani, T., Smirnov, A. et al. Reproducible mass spectrometry data processing and compound annotation in MZmine 3. *Nat. Protoc.* https://doi.org/10.1038/s41596-024-00996-y (2024).
52. Sumner, L. W. et al. Proposed minimum reporting standards for chemical analysis. *Metabolomics* **3**, 211–221 (2007).
53. Dührkop, K. et al. SIRIUS 4: a rapid tool for turning tandem mass spectra into metabolite structure information. *Nat. Methods* **16**, 299–302 (2019).
54. Dührkop, K. et al. Systematic classification of unknown metabolites using high-resolution fragmentation mass spectra. *Nat. Biotechnol.* **39**, 462–471 (2021).
55. 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).
56. 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).
57. Quinn, R. A. et al. Molecular networking as a drug discovery, drug metabolism, and precision medicine strategy. *Trends Pharmacol. Sci.* **38**, 143–154 (2017).
58. 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 Bioinforma.* **11**, 395 (2010).
59. Nguyen, L. H. & Holmes, S. Ten quick tips for effective dimensionality reduction. *PLOS Comput. Biol.* **15**, e1006907 (2019).
60. GOWER, J. C. Some distance properties of latent root and vector methods used in multivariate analysis. *Biometrika* **53**, 325–338 (1966).
61. 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).
62. Tian, M. et al. Pure ion chromatograms combined with advanced machine learning methods improve accuracy of discriminant models in LC–MS-based untargeted metabolomics. *Molecules* **26**, 2715 (2021).
63. Cacciatore, S., Tenori, L., Luchinat, C., Bennett, P. R. & MacIntyre, D. A. KODAMA: an R package for knowledge discovery and data mining. *Bioinformatics* **33**, 621–623 (2017).
64. Paliy, O. & Shankar, V. Application of multivariate statistical techniques in microbial ecology. *Mol. Ecol.* **25**, 1032–1057 (2016).
65. 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); https://doi.org/10.1007/978-1-4612-4380-9_41.
66. Desu, M. M. & Raghavarao, D. *Nonparametric Statistical Methods For Complete and Censored Data*. (CRC Press, 2003).
67. Xia, Y. & Sun, J. Hypothesis testing and statistical analysis of microbiome. *Genes Dis.* **4**, 138–148 (2017).
68. Anderson, M. J. A new method for non-parametric multivariate analysis of variance. *Austral Ecol.* **26**, 32–46 (2001).
69. Djoumbou Feunang, Y. et al. ClassyFire: automated chemical classification with a comprehensive, computable taxonomy. *J. Cheminformatics* **8**, 61 (2016).
70. Kim, H. W. et al. NPClassifier: a deep neural network-based structural classification tool for natural products. *J. Nat. Prod.* **84**, 2795–2807 (2021).
71. 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).
72. Benton, P. H. et al. An interactive cluster heat map to visualize and explore multidimensional metabolomic data. *Metabolomics. J. Metabolomic Soc.* **11**, 1029–1034 (2015).
73. 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).
74. 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).
75. 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).
76. 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).
77. Jafari, M. & Ansari-Pour, N. Why, when and how to adjust your *P* values? *Cell J. Yakhteh* **20**, 604–607 (2019).
78. Korthauer, K. et al. A practical guide to methods controlling false discoveries in computational biology. *Genome Biol.* **20**, 118 (2019).
79. Mishra, P. et al. Descriptive statistics and normality tests for statistical data. *Ann. Card. Anaesth.* **22**, 67–72 (2019).
80. Neuhaus, G. F. et al. Environmental metabolomics characterization of modern stromatolites and annotation of ibhayipeptolides. *PLoS ONE* **19**, e0303273 (2024).
81. Bolyen, E. et al. Reproducible, interactive, scalable and extensible microbiome data science using QIIME 2. *Nat. Biotechnol.* **37**, 852–857 (2019).
82. Moseley, H. N. B. Error analysis and propagation in metabolomics data analysis. *Comput. Struct. Biotechnol. J.* **4**, e201301006 (2013).
83. 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).
84. Wilkinson, M. D. et al. The FAIR guiding principles for scientific data management and stewardship. *Sci. Data* **3**, 160018 (2016).
85. Hoffmann, M. A. et al. High-confidence structural annotation of metabolites absent from spectral libraries. *Nat. Biotechnol.* **40**, 411–421 (2022).
86. Rinker, T. & Kurkiewicz, D. pacman: package management for R, version 0.5.0. https://github.com/trinker/pacman (2018).
87. Wickham, H. et al. Welcome to the Tidyverse. *J. Open Source Softw.* **4**, 1686 (2019).
88. Kluyver, T., Angerer, P. & Schulz, J. IRdisplay: 'Jupyter' display machinery. (2022).
89. Cacciatore, S., Luchinat, C. & Tenori, L. Knowledge discovery by accuracy maximization. *Proc. Natl Acad. Sci. USA* **111**, 5117–5122 (2014).
90. Kassambara, A. & Mundt, F. Factoextra: extract and visualize the results of multivariate data analyses. R package version 1.0.7. https://CRAN.R-project.org/package=factoextra (2020).
91. Oksanen, J. et al. vegan: community ecology package. R package version 2.6-4. https://doi.org/10.32614/CRAN.package.vegan (2024).
92. Gu, Z. Complex heatmap visualization. *iMeta* **1**, e43 (2022).
93. Galili, T. dendextend: an R package for visualizing, adjusting and comparing trees of hierarchical clustering. *Bioinforma. Oxf. Engl.* **31**, 3718–3720 (2015).
94. 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).
95. Archer, E. rfPermute: estimate permutation P values for random forest importance metrics. R package version 2.5.1. *CRAN* https://doi.org/10.32614/CRAN.package.rfPermute (2023).
96. Ogle, D. H., Doll, J. C., Wheeler, A. P. & Dinno, A. FSA: simple fisheries stock assessment methods. R package version 0.9.4. *CRAN* https://fishr-core-team.github.io/FSA/; https://doi.org/10.32614/CRAN.package.FSA (2023).
97. Bengtsson, H. et al. matrixStats: functions that apply to rows and columns of matrices (and to vectors). R package version 0.63.0. *CRAN* https://doi.org/10.32614/CRAN.package.matrixStats (2023).
98. Xiao, N., Cook, J., Jégousse, C., Chen, H. & Li, M. ggsci: scientific journal and sci-fi themed color palettes for 'ggplot2'. R package version 3.0. *CRAN* https://doi.org/10.32614/CRAN.package.ggsci (2023).
99. Wilke, C. O. cowplot: streamlined plot theme and plot annotations for 'ggplot2'. R package version 1.1.1. *CRAN* https://doi.org/10.32614/CRAN.package.cowplot (2020).
100. Wickham, H. et al. svglite: an 'SVG' graphics device. R package version 2.1.1. *CRAN* https://doi.org/10.32614/CRAN.package.svglite (2023).
101. 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).
102. Burton, L. et al. Instrumental and experimental effects in LC–MS-based metabolomics. *J. Chromatogr. B* **871**, 227–235 (2008).
103. Gregori, J. et al. Batch effects correction improves the sensitivity of significance tests in spectral counting-based comparative discovery proteomics. *J. Proteom.* **75**, 3938–3951 (2012).
104. 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).
105. Johnson, W. E., Li, C. & Rabinovic, A. Adjusting batch effects in microarray expression data using empirical Bayes methods. *Biostatistics* **8**, 118–127 (2007).
106. 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).
107. Wehrens, R. et al. Improved batch correction in untargeted MS-based metabolomics. *Metabolomics* **12**, 88 (2016).
108. 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).
109. 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). *Analyst* **140**, 7810–7817 (2015).
110. 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).
111. 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).
112. Dmitrenko, A., Reid, M. & Zamboni, N. Regularized adversarial learning for normalization of multi-batch untargeted metabolomics data. *Bioinformatics* **39**, btad096 (2023).
113. 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).
114. Liu, Q. et al. Addressing the batch effect issue for LC/MS metabolomics data in data preprocessing. *Sci. Rep.* **10**, 13856 (2020).

115. 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).
116. 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).
117. Wulff, J. E. & Mitchell, M. W. A comparison of various normalization methods for LC/MS metabolomics data. *Adv. Biosci. Biotechnol.* **9**, 339–351 (2018).
118. 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).
119. 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).
120. Morgan, M. & Ramos, M. BiocManager: access the bioconductor project package repository. (2023).
121. 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).
122. Wilkinson, L. & Friendly, M. The history of the cluster heat map. *Am. Stat.* **63**, 179–184 (2009).
123. Wu, W. & Noble, W. S. Genomic data visualization on the Web. *Bioinformatics* **20**, 1804–1805 (2004).
124. Griffiths, E. T. et al. Detection and classification of narrow-band high frequency echolocation clicks from drifting recorders. *J. Acoust. Soc. Am.* **147**, 3511–3522 (2020).
125. Liu, S. et al. Comammox biogeography subject to anthropogenic interferences along a high-altitude river. *Water Res.* **226**, 119225 (2022).
126. Breiman, L. Random Forests. *Mach. Learn.* **45**, 5–32 (2001).
127. Liaw, A. & Wiener, M. Classification and regression by randomForest. *R News* **2**, 18–22 (2002); https://journal.r-project.org/articles/RN-2002-022/RN-2002-022.pdf.
128. Robinson, D. et al. broom: convert statistical objects into tidy tibbles. *CRAN* https://doi.org/10.32614/CRAN.package.broom (2023).
129. Vinaixa, M. et al. A Guideline to univariate statistical analysis for LC/MS-based untargeted metabolomics-derived data. *Metabolites* **2**, 775–795 (2012).
130. Ostertagová, E., Ostertag, O. & Kováč, J. Methodology and application of the Kruskal–Wallis test. *Appl. Mech. Mater.* **611**, 115–120 (2014).
131. 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).
132. Giacomoni, F. et al. Workflow4Metabolomics: a collaborative research infrastructure for computational metabolomics. *Bioinformatics* **31**, 1493–1495 (2015).
133. Kontou, E. E. et al. UmetaFlow: an untargeted metabolomics workflow for high-throughput data processing and analysis. *J. Cheminformatics* **15**, 52 (2023).
134. 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).
135. Chong, J. & Xia, J. MetaboAnalystR: an R package for flexible and reproducible analysis of metabolomics data. *Bioinformatics* **34**, 4313–4314 (2018).
136. Pang, Z. & Xia, J. LC–MS/MS raw spectral data processing. https://www.metaboanalyst.ca/resources/vignettes/LCMSMS_Raw_Spectral_Processing.html (2024).
137. Tiffany, C. R. & Bäumler, A. J. omu, a metabolomics count data analysis tool for intuitive figures and convenient metadata collection. *Microbiol. Resour. Announc.* **8**, e00129-19 (2019).
138. Han, X. & Liang, L. metabolomicsR: a streamlined workflow to analyze metabolomic data in R. *Bioinforma. Adv.* **2**, vbac067 (2022).
139. 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).
140. Thévenot, E. A., Roux, A., Xu, Y., Ezan, E. & Junot, C. Analysis of the human adult urinary metabolome variations with age, body mass index, and gender by implementing a comprehensive workflow for univariate and OPLS statistical analyses. *J. Proteome Res.* **14**, 3322–3335 (2015).
141. 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).
142. 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).
143. Ivanisevic, J. & Want, E. J. From samples to insights into metabolism: uncovering biologically relevant information in LC–HRMS metabolomics data. *Metabolites* **9**, 308 (2019).
144. 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).
145. Martínez-Sena, T. et al. Monitoring of system conditioning after blank injections in untargeted UPLC–MS metabolomic analysis. *Sci. Rep.* **9**, 9822 (2019).
146. Raynie, D. The vital role of blanks in sample preparation. *LCGC N. Am.* **36**, 494–497 (2018).
147. 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).
148. Livera, A. M. D. et al. Statistical methods for handling unwanted variation in metabolomics data. *Anal. Chem.* **87**, 3606–3615 (2015).
149. 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).
150. 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).
151. Schiffman, C. et al. Filtering procedures for untargeted LC–MS metabolomics data. *BMC Bioinforma.* **20**, 334 (2019).
152. 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).
153. Wei, R. et al. Missing value imputation approach for mass spectrometry-based metabolomics data. *Sci. Rep.* **8**, 663 (2018).
154. 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).
155. 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).
156. Scholz, M., Gatzek, S., Sterling, A., Fiehn, O. & Selbig, J. Metabolite fingerprinting: detecting biological features by independent component analysis. *Bioinformatics* **20**, 2447–2454 (2004).
157. Deininger, S.-O. et al. Normalization in MALDI-TOF imaging datasets of proteins: practical considerations. *Anal. Bioanal. Chem.* **401**, 167–181 (2011).
158. Qannari, E. M., Wakeling, I., Courcoux, P. & MacFie, H. J. H. Defining the underlying sensory dimensions. *Food Qual. Prefer.* **11**, 151–154 (2000).
159. Khalheim, O. M. Scaling of analytical data. *Anal. Chim. Acta* **177**, 71–79 (1985).
160. 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).
161. 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).
162. 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).
163. 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).
164. Lozupone, C. & Knight, R. UniFrac: a new phylogenetic method for comparing microbial communities. *Appl. Environ. Microbiol.* **71**, 8228–8235 (2005).
165. Brejnrod, A. et al. Implementations of the chemical structural and compositional similarity metric in R and Python. Preprint at *bioRxiv* https://doi.org/10.1101/546150 (2019).
166. Tripathi, A. et al. Chemically informed analyses of metabolomics mass spectrometry data with Qemistree. *Nat. Chem. Biol.* **17**, 146–151 (2021).
167. Ramette, A. Multivariate analyses in microbial ecology. *FEMS Microbiol. Ecol.* **62**, 142–160 (2007).
168. Koenig, J. E. et al. Succession of microbial consortia in the developing infant gut microbiome. *Proc. Natl Acad. Sci.* **108**, 4578–4585 (2011).
169. Archer, F. I., Martien, K. K. & Taylor, B. L. Diagnosability of mt DNA with random forests: using sequence data to delimit subspecies. *Mar. Mammal. Sci.* **33**, 101–131 (2017).
170. Breiman, L. Out-of-bag estimation. Technical report 1-13 (Statistics Department, University of California Berkeley, 1996); https://www.stat.berkeley.edu/pub/users/breiman/OOBestimation.pdf.
171. Strobl, C., Boulesteix, A.-L., Kneib, T., Augustin, T. & Zeileis, A. Conditional variable importance for random forests. *BMC Bioinforma.* **9**, 307 (2008).
172. Archer, K. J. & Kimes, R. V. Empirical characterization of random forest variable importance measures. *Comput. Stat. Data Anal.* **52**, 2249–2260 (2008).
173. Riffenburgh, R. H. & Gillen, D. L. *Statistics in Medicine* (Academic Press, 2020).
174. Sato, T. Type I and type II error in multiple comparisons. *J. Psychol.* **130**, 293–302 (1996).
175. 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).
176. Abdi, H. & Williams, L. Newman–Keuls test and Tukey test. *Encycl. Res. Des.* (2010).
177. Hecke, T. V. Power study of anova versus Kruskal–Wallis test. *J. Stat. Manag. Syst.* **15**, 241–247 (2012).
178. Dinno, A. Nonparametric pairwise multiple comparisons in independent groups using Dunn's test. *Stata J. Promot. Commun. Stat. Stata* **15**, 292–300 (2015).

Acknowledgements

We thank G. Caporaso for guidance on preparing the QIIME2 plugins. D.P., C.M. and H.L. were supported by the Deutsche Forschungsgemeinschaft (DFG) through the CMFI Cluster of Excellence (EXC 2124) and D.P. and C.M., were supported by the DFG through the Collaborative Research Center CellMap (TRR 261). K.D. was supported by the DFG (BO 1910/23). P.S. was supported by the European Union's Horizon Europe research and innovation programme through a Marie Skłodowska-Curie fellowship no. 101108450 MeStaLeM. T.P. 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. T.D. was supported by the MSCA Fellowships CZ (OP JAK) grant CZ.02.01.01/00/22_010/0002733. M.W. was supported by the National Institutes of Health (NIH) with grants 1U24DK133658-01, NIH 1R03DE032437-01 and UC Riverside startup funding and was partially supported by the US Department of Energy Joint Genome Institute

(https://ror.org/04xm1d337), a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy operated under contract DE-AC02-05CH11231. E.E.K. was supported by grants from the Novo Nordisk Foundation [NNF20CC0035580, NNF16OC0021746]. Y.W. was supported by NIH 1R03DE032437-01. C.B. was supported by the Czech Academy of Sciences (CAS PPLZ) L200552251. F.O. was supported by FAPESP 2022/14603-8. J.B.'s work was carried out as part of the German Center for Infection Research (DZIF) project 09.720. We thank L. Lo Presti for critical reading of the manuscript.

Author contributions

A.K.P.S., F.O., F.R., M.E. and D.P. conceptualized the protocol. Y.E., S.Z., J.S. and R.S. advised on the concept and statistical test. A.K.P.S., A.W., F.O., F.R., M.N., J.B., E.E.K., J.E., A.P., C.G.M., S.F., N.C., Y.W., M.D., J.S., M.W. and M.E. wrote code. A.K.P.S., A.W. and M.W. developed and deployed the web application. R.S., A.T.A. and D.P. collected the water samples. D.P. extracted the samples and acquired the LC–MS/MS data. A.K.P.S., A.W., F.O., F.R., M.N., J.B., J.J.K., E.E.K., J.E., A.P., C.G.M., S.F., M.R.A., T.P., N.C., M.P., C.B., B.C., A.M.C.R., A.C., F.D., K.D., Y.E., C.G., L.G.G., M.H., S.H., S.K., A.K., M.C.M.K., K.M., S.P., P.W.P., T.S., K.S.L., P.S., S.T., G.A.V., B.C.W., S.X., M.T.Y., S.Z., M.D., C.B., H.L., C.M., J.J.J.v.d.H., T.D., P.C.D., J.S., R.S., A.T.A., M.E. and D.P. tested the protocol, code and app. C.B., J.J.J.v.d.H., T.P., M.W., A.T.A., M.E. and D.P. supervised students and researchers. M.W., A.A., M.E. and D.P. supervised the project. A.K.P.S., M.N., J.B., J.J.K., E.E.K., A.P., S.F., T.P., A.T.A. and D.P. wrote the manuscript and supplemental information. F.O., F.R., J.E., C.G.M., M.R.A., N.C., M.P., K.D., Y.E., L.G.G., M.H., S.H., P.S., G.A.V., S.Z., J.J.J.v.d.H., T.D., T.P., P.C.D., J.S., R.S., M.W. and M.E. edited and provided critical feedback on the first draft. All authors edited and approved the final draft.

Competing interests

J.J.J.v.d.H. is currently a member of the Scientific Advisory Board of Naicons Srl., Milano, Italy, and is consulting for Corteva Agriscience. P.C.D. is a scientific advisor and holds equity to Cybele and a cofounder, advisor, and holds equity in Ometa, Arome and Enveda with prior approval by UC-San Diego and consulted in 2023 for DSM animal health. M.W. is the founder of Ometa Labs. S.H., T.P. and R.S. are cofounders of mzio GmbH.

Additional information

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s41596-024-01046-3.

Correspondence and requests for materials should be addressed to Madeleine Ernst or Daniel Petras.

Peer review information *Nature Protocols* thanks Vinny Davies, Charlotte Simmler and the other, anonymous, reviewer(s) for their contribution to the peer review of this work.

Related links

Key references using this protocol

Nothias, L.-F. et al. *Nat. Methods* **17**, 905–908 (2020): https://doi.org/10.1038/s41592-020-0933-6
Heuckeroth, S. et al. *Nat. Protoc.* (2024): https://doi.org/10.1038/s41596-024-00996-y
Petras, D. et al. *Chemosphere* **271**, 129450 (2021): https://doi.org/10.1016/j.chemosphere.2020.129450
Wegley Kelly, L. et al. *Proc. Natl Acad. Sci. USA* **119**, e2110283119 (2022): https://doi.org/10.1073/pnas.2110283119
Neuhaus, G. F. et al. *PLOS One* **19**, e0303273 (2024): https://doi.org/10.1371/journal.pone.0303273

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

[1]Virtual Multi-Omics Laboratory, The Internet, Riverside, CA, USA. [2]University of Tübingen, Interfaculty Institute of Microbiology and Infection Medicine, Tübingen, Germany. [3]Applied Bioinformatics, Department of Computer Science, University of Tübingen, Tübingen, Germany. [4]Section for Clinical Mass Spectrometry, Danish Center for Neonatal Screening, Department of Congenital Disorders, Statens Serum Institut, Copenhagen S, Denmark. [5]Leibniz Institute DSMZ–German Collection of Microorganisms and Cell Cultures, Braunschweig, Germany. [6]German Center for Infection Research, Partner Site Braunschweig-Hannover, Braunschweig, Germany. [7]Department of Biochemistry and Microbiology, Rhodes University, Makhanda, South Africa. [8]The Novo Nordisk Foundation for Biosustainability, Technical University of Denmark, Kongens Lyngby, Denmark. [9]Department of Chemistry and Biochemistry, University of Denver, Denver, CO, USA. [10]Boyce Thompson Institute and Department of Chemistry and Chemical Biology, Cornell University, Ithaca, NY, USA. [11]Universidad EAFIT, Medellín, Antioquia, Colombia. [12]Bigelow Laboratory for Ocean Sciences, East Boothbay, ME, USA. [13]School of Marine Sciences, Darling Marine Center, University of Maine, Walpole, ME, USA. [14]Department of Environmental Science, Aarhus University, Roskilde, Denmark. [15]National Center for Genetic Engineering and Biotechnology, National Science and Technology Development Agency, Thailand Science Park, Pathum Thani, Thailand. [16]Department of Computer Science, University of California Riverside, Riverside, CA, USA. [17]Department of Food Chemistry and Toxicology, University of Vienna, Vienna, Austria. [18]Institute of Organic Chemistry and Biochemistry of the Czech Academy of Sciences, Prague, Czech Republic. [19]Laboratorio de Microbiología Molecular y Biotecnología Ambiental, Centro de Biotecnología DAL, Universidad Técnica Federico Santa María, Valparaíso, Chile. [20]Skaggs School of Pharmacy and Pharmaceutical Sciences, University of California San Diego, San Diego, CA, USA. [21]Department of Biotechnology, Engineering School of Lorena, University of São Paulo, Lorena, São Paulo, Brazil. [22]Department of Bioinformatics, University of Jena, Jena, Germany. [23]Department of Environmental Systems Analysis, University of Tübingen, Tübingen, Germany. [24]Leibniz Institute of Freshwater Ecology and Inland Fisheries,

Berlin, Germany. [25]Institute of Inorganic and Analytical Chemistry, University of Münster, Münster, Germany. [26]Department of Ecology, Behavior and Evolution, University of California San Diego, San Diego, CA, USA. [27]Department of Analytical Chemistry, University of Vienna, Vienna, Austria. [28]Bioinformatics Group, Wageningen University and Research, Wageningen, the Netherlands. [29]Department of Biochemistry, University of California Riverside, Riverside, CA, USA. [30]Collaborative Mass Spectrometry Innovation Center, Skaggs School of Pharmacy and Pharmaceutical Sciences, University of California San Diego, San Diego, CA, USA. [31]Helmholtz Institute for Pharmaceutical Research Saarland, Helmholtz Centre for Infection Research, Saarbrücken, Germany. [32]Saarland University, Saarbrücken, Germany. [33]Department of Biochemistry, University of Johannesburg, Johannesburg, South Africa. [34]Department of Nutrition, Exercise and Sports, University of Copenhagen, Frederiksberg C, Denmark.

nature portfolio

Corresponding author(s): Madeleine Ernst, Daniel Petras

Last updated by author(s): 2024/05/27

Reporting Summary

Nature Portfolio wishes to improve the reproducibility of the work that we publish. This form provides structure for consistency and transparency in reporting. For further information on Nature Portfolio policies, see our Editorial Policies and the Editorial Policy Checklist.

Statistics

For all statistical analyses, confirm that the following items are present in the figure legend, table legend, main text, or Methods section.

n/a	Confirmed
☒	☐ The exact sample size (*n*) for each experimental group/condition, given as a discrete number and unit of measurement
☐	☒ A statement on whether measurements were taken from distinct samples or whether the same sample was measured repeatedly
☐	☒ The statistical test(s) used AND whether they are one- or two-sided *Only common tests should be described solely by name; describe more complex techniques in the Methods section.*
☐	☒ A description of all covariates tested
☐	☒ A description of any assumptions or corrections, such as tests of normality and adjustment for multiple comparisons
☐	☒ A full description of the statistical parameters including central tendency (e.g. means) or other basic estimates (e.g. regression coefficient) AND variation (e.g. standard deviation) or associated estimates of uncertainty (e.g. confidence intervals)
☒	☐ For null hypothesis testing, the test statistic (e.g. *F*, *t*, *r*) with confidence intervals, effect sizes, degrees of freedom and *P* value noted *Give P values as exact values whenever suitable.*
☒	☐ For Bayesian analysis, information on the choice of priors and Markov chain Monte Carlo settings
☒	☐ For hierarchical and complex designs, identification of the appropriate level for tests and full reporting of outcomes
☒	☐ Estimates of effect sizes (e.g. Cohen's *d*, Pearson's *r*), indicating how they were calculated

Our web collection on statistics for biologists contains articles on many of the points above.

Software and code

Policy information about availability of computer code

Data collection	Raw and processed data are available through the MassIVE repository: MSV000082312 and MSV000085786 https://massive.ucsd.edu/ProteoSAFe/dataset.jsp?task=8a8139d9248b43e0b0fda17495387756 https://massive.ucsd.edu/ProteoSAFe/dataset.jsp?task=c8411b76f30a4f4ca5d3e42ec13998dc and Zenodo (https://zenodo.org/records/10051610) with the following doi https://doi.org/10.5281/zenodo.10051610.
Data analysis	All code and software are available through GitHub under the following link https://github.com/Functional-Metabolomics-Lab/FBMN-STATS. The web application can be accessed at https://fbmn-statsguide.gnps2.org/. Downloadable Windows executables of the web app is available from https://www.functional-metabolomics.com/resources. All the codes are deposited on Zenodo with the following doi: https://doi.org/10.5281/zenodo.11350947.

For manuscripts utilizing custom algorithms or software that are central to the research but not yet described in published literature, software must be made available to editors and reviewers. We strongly encourage code deposition in a community repository (e.g. GitHub). See the Nature Portfolio guidelines for submitting code & software for further information.

Data

Policy information about availability of data

All manuscripts must include a data availability statement. This statement should provide the following information, where applicable:

- Accession codes, unique identifiers, or web links for publicly available datasets
- A description of any restrictions on data availability
- For clinical datasets or third party data, please ensure that the statement adheres to our policy

Provide your data availability statement here.

Human research participants

Policy information about studies involving human research participants and Sex and Gender in Research.

Reporting on sex and gender	N/A
Population characteristics	N/A
Recruitment	N/A
Ethics oversight	N/A

Note that full information on the approval of the study protocol must also be provided in the manuscript.

Field-specific reporting

Please select the one below that is the best fit for your research. If you are not sure, read the appropriate sections before making your selection.

☐ Life sciences ☐ Behavioural & social sciences ☒ Ecological, evolutionary & environmental sciences

For a reference copy of the document with all sections, see nature.com/documents/nr-reporting-summary-flat.pdf

Ecological, evolutionary & environmental sciences study design

All studies must disclose on these points even when the disclosure is negative.

Study description	This is a protocols paper, that introduces the statistical analysis of feature-based molecular networking results, with published / public data for test purposes
Research sample	N/A
Sampling strategy	N/A
Data collection	N/A
Timing and spatial scale	N/A
Data exclusions	N/A
Reproducibility	N/A
Randomization	N/A
Blinding	N/A

Did the study involve field work? ☐ Yes ☒ No

Reporting for specific materials, systems and methods

We require information from authors about some types of materials, experimental systems and methods used in many studies. Here, indicate whether each material, system or method listed is relevant to your study. If you are not sure if a list item applies to your research, read the appropriate section before selecting a response.

Materials & experimental systems

n/a	Involved in the study
☒	☐ Antibodies
☒	☐ Eukaryotic cell lines
☒	☐ Palaeontology and archaeology
☒	☐ Animals and other organisms
☒	☐ Clinical data
☒	☐ Dual use research of concern

Methods

n/a	Involved in the study
☒	☐ ChIP-seq
☒	☐ Flow cytometry
☒	☐ MRI-based neuroimaging