MassCube: a Python framework for end-to-end metabolomics data processing from raw files to phenotype classifiers

Huaxu Yu[1], Jun Ding[2], Tong Shen[1], Min Liu[1], Yuanyue Li[1], Oliver Fiehn[1*]

Affiliations:

[1] West Coast Metabolomics Center, University of California Davis, Davis, CA, 95616, USA

[2] Wuhan Botanical Garden, Chinese Academy of Sciences No. 201 Jiufeng 1 Road, East Lake High-Tech Development Zone, Wuhan, Hubei, P. R. China

* Corresponding author:

Oliver Fiehn: ofiehn@ucdaivs.edu

Supplementary materials:

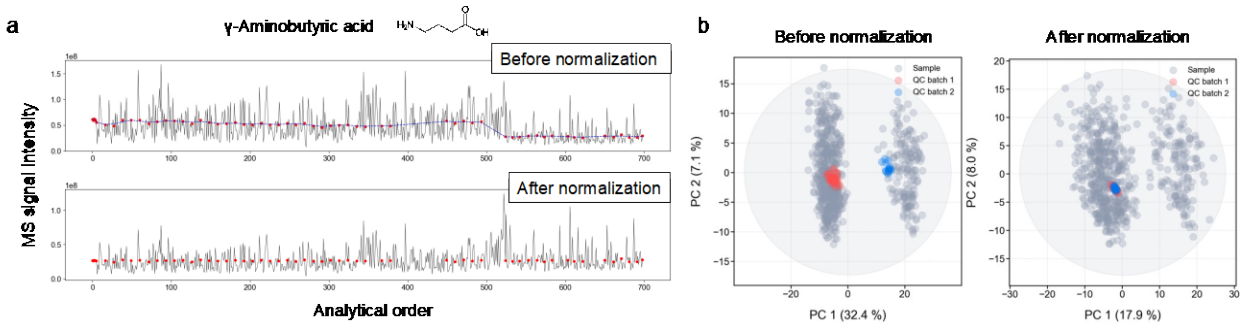
Supplementary table 8: Retention time correction validation results

Supplementary table 9: Source data for Fig. 2 in the main text

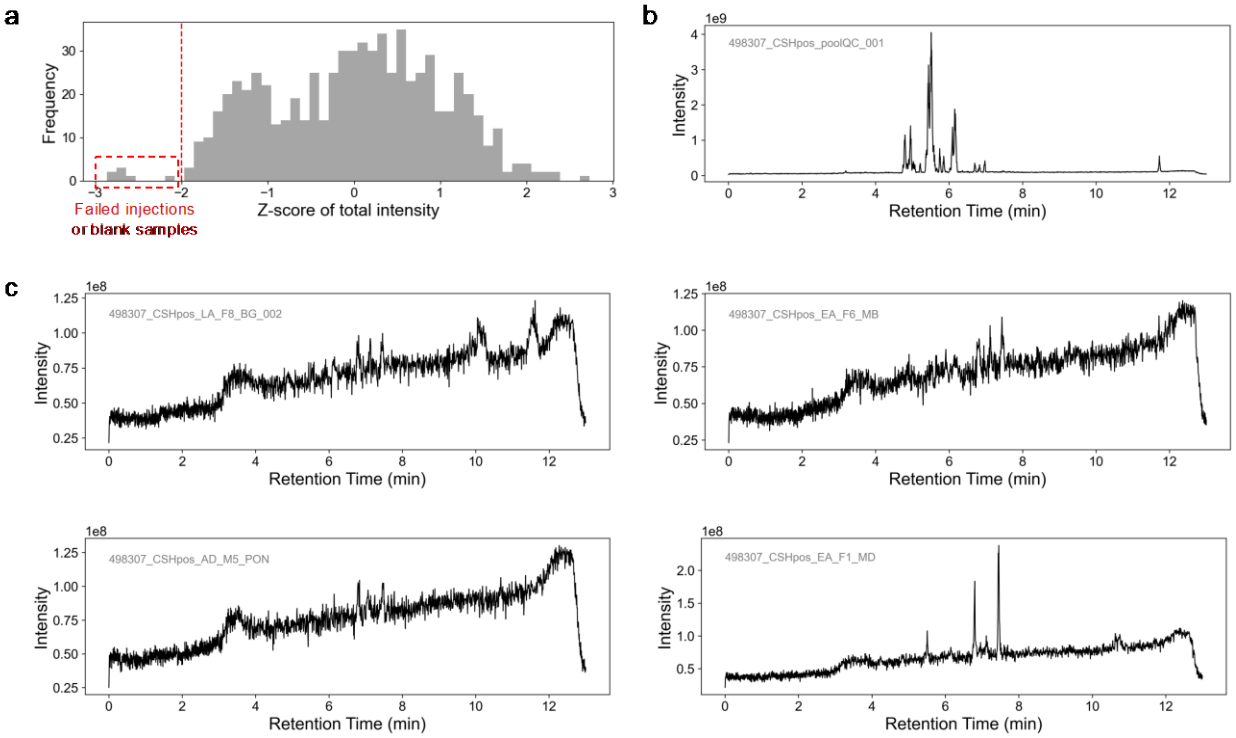
Supplementary table 10: Source data for Fig. 3 in the main text

Supplementary table 11: Source data for Fig. 4 in the main text

Supplementary table 12: Source data for Fig. 5 in the main text



Supplementary Figure 1. **Validating normalization of systematic mass spectrometry signal drift.** a) An example of the signal drift of the metabolite gamma-aminobutyric acid before and after normalization. The black line represents the MS signal intensity in individual samples plotted by analytical order. Red dots indicate quality control (QC) samples, while the dashed blue line shows the LOWESS calibration curve generated from QC samples. b) Principal component analysis results before and after normalization. The clustering of QC samples illustrates the reduced batch effect post-normalization. Data were obtained from the aging mouse brain dataset in case study 1.



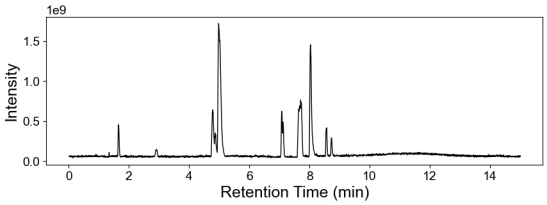
Supplementary Figure 2. **Detected failed sample injections by MassCube.** a) Distribution of Z-scores computed using total intensities from detected features. b) Base peak chromatogram of a normal sample injection. c) Base peak chromatograms of four detected failed sample injections. Data were obtained from the aging mouse brain dataset in case study 1, analyzed in reverse phase positive mode.

a Read raw MS data to MSData object

```
data = read_raw_file_to_obj("sample.mzML")
```

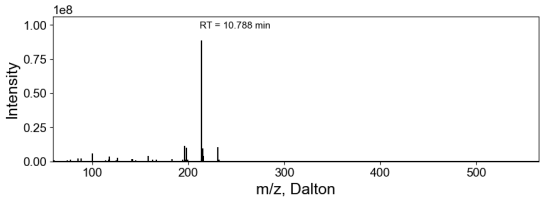
b Example 1 | plot base peak chromatogram

```
data.plot_bpc()
```



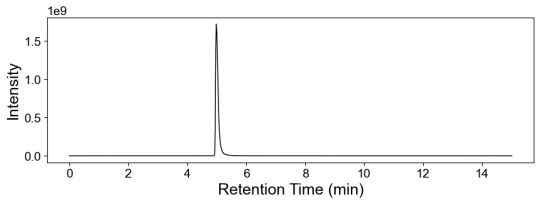
c Example 2 | plot scan with # 6440

```
data.scans[6440]
```



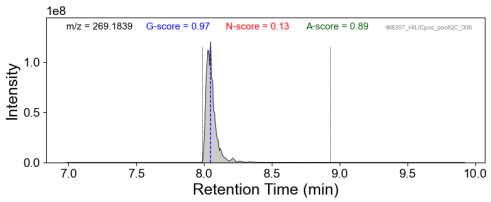
d Example 3 | plot extract ion chromatogram for m/z = 113.1631

```
data.plot_eic(mz=113.1631
)
```



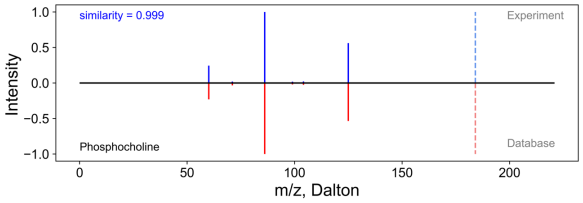
e Example 4 | plot a feature in data

```
plot_roi(data, roi=r)
```



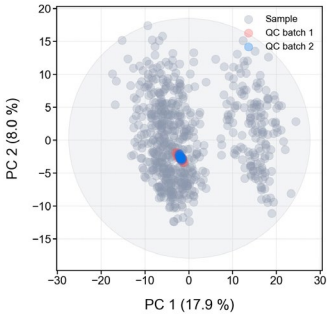
f Example 5 | plot a MS/MS matching result

```
mirror_ms2(precursor_mz1, precursor_mz2,
peaks1, peaks2, annotation)
```



g Example 6 | make PCA plot

```
pca_analysis(data
_array,
individual_sample
_groups)
```



Supplementary Figure 3. **Examples of data visualization in MassCube.** a) Code block to read raw MS data in MassCube for visualization. Examples include plotting the base peak chromatogram (b), a single scan (c), extracted ion chromatogram (d), a detected feature (e), an MS/MS spectral matching result (f), and a principal component analysis plot (g). Code to generate the plots using the *masscube* Python package is shown in blue.

Supplementary Note 1. Metadata management in MassCube

MassCube provides a diverse range of data processing functions and modules to support various application-oriented data processing tasks. Therefore, it is crucial to record all data processing steps in sequence, along with the parameters used, to ensure that the entire workflow can be traced and reproduced. We propose a stepwise framework to store the complete workflow, following the structure outlined below:

```
medadata = [
{
"name": "overview",
"layer": 0,
"packages": [
{"name": "masscube", "version": version("masscube")},
{"name": "numpy", "version": version("numpy")},
   …
],
"start_time": (time when data analysis begins)
…
},
{
"name": " feature_detection",
"layer ": 1,
(parameters): (value),
….
},
….
]
```

where the key "name" describes the name of each step, "layer" controls the order in which the steps are executed, and (parameters):(value) records the step-specific data processing parameters. This structured approach ensures clarity and reproducibility of the data processing workflow within MassCube.

Supplementary Note 2. LC-MS analysis of human plasma samples of Alzheimer's Disease patients

473 plasmas were analyzed by both Exploris 240 Orbitrap and Astral Orbitrap (Thermo Scientific, San Jose) from an Alzheimer's patients exposome cohort from Prof. Rima Kaddoura-Daouk at Duke University. Briefly, 20 µL of plasma was extracted by Matyash biphasic extraction. The lower aqueous phase was transferred to two aliquots which both were dried down and resuspended with 100 µL of ACN:Water (80:20) with isotope-labeled internal standards. One aliquot was analyzed by Exploris 240 and the other aliquot was analyzed by Astral. 3 µL of the polar metabolite extract was injected into HILIC-MS. The HILIC column was ACQUITY Premier BEH Amide Column, 1.7 µm, 2.1 mm x 50 mm. Mobile phases A and B were water and 95% ACN both with 10mM ammonium formate and 0.125% formic acid. The LC gradient started with 100% of B and decreased to 30% B from 0.5 to 2.55 min, followed by 3.15 to 3.8 min of equilibration back to 100% of B. The LC method was kept the same between Exploris 240 and Astral. The polar metabolites were analyzed in both positive and negative ionization modes. In Exploris 240, every sample was acquired by MS1 full scan (60-900 mass range, 60,000 mass resolution, 1e6 AGC target) and top-2 DDA MS2 (15,000 mass resolution, 10^5 AGC target, NCE 30-50-80%). 8 rounds of iterative exclusion were acquired on the pooled QC sample by automated Acquire X. In the Astral system, every sample was analyzed with full scan MS1 with Orbitrap analyzer (60-900 mass range, 60,000 mass resolution, 1e6 AGC target) and MS/MS scans with Astral analyzer (15,000 mass resolution, 1% of 1e5 AGC target, NCE of 40%). With the same 0.2 msec cycle time, the MS/MS by Astral with 1% of 1e5 AGC target was approximately equivalent to top-35 data-dependent acquisition. Pooled quality control, reference material NIST 1950 plasma, and blanks were acquired every 10 sample injections for quality control purposes.

Supplementary Note 3. Failed sample injection detection algorithm

Failed sample injections can occur for various reasons, including insufficient sample volume, uncalibrated needle position, and improper insert position in the vial. When an injection fails, the sample should not be used for downstream data normalization or statistical analysis. It can be anticipated that if a quality control sample fails, intensity normalization may be compromised due to extremely low intensity values in that sample. MassCube automatically evaluates the data and identifies failed sample injections using an outlier detection algorithm.

Individual data files were first processed in MassCube for feature detection. Using all the detected features, MassCube computed the total intensity (using peak height by default) for all the n files in a study, resulting in the set $\{Int_1, \ldots Int_n\}$. Z-scores were subsequently calculated using the total intensity values, yielding $\{z_1, \ldots, z_n\}$. A file *i* is considered an outlier when

$$z_i > zscore_{tol}$$

where the tolerance of Z-score was set to 2 by default. Importantly, blank samples, which typically exhibit significantly lower total intensity, are not considered as failed injections.