Package 'postvocs'

November 10, 2026

Title Post-Processing Tools for GC-MS Volatile Organic Compound Data

Version 0.2.6

Description Provides functions for processing Shimadzu gas chromatography-mass spectrometry (GC-MS) exported text files, extracting Chemical Abstracts Service (CAS) numbers and peak areas, building abundance matrices, annotating compounds, and screening based on occurrence frequency. The package is designed for organizing, identifying, and screening volatile compounds in metabolomics and environmental studies.

Depends R (>= 4.5.0)

Imports dplyr, tidyr, openxlsx, readxl, rlang, webchem, tools

License GPL (>= 3)

URL https://github.com/HanXT97/postvocs

BugReports https://github.com/HanXT97/postvocs/issues

Language en-US

Encoding UTF-8

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Xiaotao Han [aut, cre]

Maintainer Xiaotao Han <postvocs.r@gmail.com>

Repository CRAN

Date/Publication 2026-11-10 10:10:02 UTC

Contents

annotate_compounds 2
batch_process_gcms 4
build_cas_abundance 5
extract_peak_areas 6
filter_by_frequency 8

process_gcms_txt 9
save_gcms_results 11
save_postvocs_results 12

Index **14**

`annotate_compounds` *Annotate CAS numbers with chemical information*

Description

This function takes a set of CAS numbers (from an abundance matrix) and retrieves chemical names and properties from user-provided libraries or from online databases (via ‘webchem‘). It supports caching and automatic rate-limiting to comply with PubChem API constraints.

Usage

```
annotate_compounds(
  abundance_data,
  lib_source = c("auto", "user", "webchem"),
  user_lib = NULL,
  webchem_config = list(rate_limit = 5, chunk_size = 100, cache_file = NULL),
  use_cache = FALSE,
  force_retrieve = FALSE
)
```

Arguments

`abundance_data` Either:

- A data.frame with a first column named "CAS"
- A character string path to a CSV file with the same structure
- A character string path to an Excel file (.xlsx or .xls) with the same structure

`lib_source` Character. Source of chemical information:

- ‘"user"‘ - only use ‘user_lib‘; CAS not found become ‘NA‘.
- ‘"webchem"‘ - only query online via ‘webchem‘.
- ‘"auto"‘ - first try ‘user_lib‘, then webchem for missing ones.

Default is ‘"auto"‘.

`user_lib` Optional data.frame with at least two columns: ‘CAS‘ and ‘Name‘. Used when ‘lib_source = "user"‘ or ‘"auto"‘. Must be provided if ‘lib_source = "user"‘.

`webchem_config` A list controlling online queries:

- ‘rate_limit‘ - maximum requests per second (default 5).
- ‘chunk_size‘ - number of CAS per batch (default 100).
- ‘cache_file‘ - path to RDS cache file. If not provided and ‘use_cache = TRUE‘, defaults to the user cache directory (e.g., ‘tools::R_user_dir("postvocs", "cache")/chem_cache.rds‘).

`use_cache` Logical. Whether to use caching. Default is 'FALSE'. If 'FALSE', no cache file is read or written, and 'force_retrieve' is ignored. If 'TRUE', caching is enabled; the cache file path is printed to the console.

`force_retrieve` Logical. If 'TRUE' and 'use_cache = TRUE', ignore existing cache and query all CAS again. Default 'FALSE'. Has no effect when 'use_cache = FALSE'.

Details

The function automatically determines the batch strategy based on the number of CAS to query:

- 1-50: direct individual queries (no chunking).
- 51-1000: chunked queries using 'chunk_size'.
- 1001-5000: chunked queries with smaller chunk size (200) and extra delay.
- > 5000: a warning is issued recommending the PubChem PUG Download service; the function will still attempt chunked queries but may be slow or rate-limited.

A persistent cache is stored at the location given by 'cache_file' (or the default user cache directory) only when 'use_cache = TRUE'. The cache is a named list where each key is a CAS number and the value is a list of chemical properties, including a 'status' field ('"success"' or '"not_found"'). Cache is updated after each query when enabled.

The 'Annotation_Source' column indicates the origin of the compound name:

- '"user_lib"' - from the user-provided library.
- '"web"' - newly queried from webchem in this session.
- '"web(cache)"' - retrieved from the cache (previous webchem query).

CAS numbers are automatically cleaned before processing: leading/trailing single quotes are removed and all spaces are stripped.

Value

A list with two components:

`annotation` A data.frame containing all CAS numbers and their retrieved information (ID, CAS, Name, MF, MW, IUPAC_Name, SMILES, InChIKey, InChI, QueryDate, Status, Source).

`abundance_updated` The original abundance data.frame with two additional columns: 'Compound_Name' and 'Annotation_Source', inserted after the 'CAS' column.

Examples

```
# Create a small abundance matrix
ab <- data.frame(
  CAS = c("64-17-5", "67-64-1", "75-05-8"),
  Sample1 = c(100, 200, 300),
  Sample2 = c(150, 250, 350)
)
```

```
# Create a user-provided compound library
user_lib <- data.frame(
  CAS = c("64-17-5", "67-64-1"),
  Name = c("Ethanol", "Acetone"),
  stringsAsFactors = FALSE
)

# Annotate using the user library (no network required)
res <- annotate_compounds(ab, lib_source = "user", user_lib = user_lib)

# View annotation table
head(res$annotation)

# Updated abundance matrix with Compound_Name and Annotation_Source
head(res$abundance_updated)
```

`batch_process_gcms` *Batch process GC-MS TXT files in a directory with sample name mapping*

Description

Processes all TXT files in a given directory using `process_gcms_txt()`, maps the raw file names to user-defined sample names via an Excel mapping file, and returns two lists: one containing all PeakTables and one containing all SearchResults, each named by the mapped sample name.

Usage

```
batch_process_gcms(
  txt_dir,
  sample_file,
  sheet = 1,
  pattern = "\\.txt$",
  encoding = "UTF-8",
  ...
)
```

Arguments

`txt_dir`	Character. Path to the directory containing the TXT files.
`sample_file`	Character. Path to an Excel file with two columns: `FileID` (raw TXT file identifier without extension, matching the TXT filename exactly) and `SampleName` (user-defined sample name).
`sheet`	Integer or character. Sheet name or index in the Excel file. Default is `1` (first sheet).
`pattern`	Character. Regular expression for file pattern; default `"\.txt$"`.

encoding	Character. Encoding of the input TXT files. Default is "UTF-8". Change to "GBK" or "latin1" if the files contain non-UTF-8 characters (e.g., in NIST library paths).
...	Additional arguments passed to process_gcms_txt() (e.g., debug = TRUE). If debug = TRUE is passed, the detailed output from process_gcms_txt will be printed.

Value

A list with four components:

PeakTables	A named list of data frames, each representing the PeakTable of a sample. Names are "SampleName_PeakTable".
SearchResults	A named list of data frames, each representing the SearchResults of a sample. Names are "SampleName_SearchResults".
failed_files	Character vector of raw file names that failed to process.
summary	A data frame with total, success, and failed counts.

Examples

```
# Get paths to example data
txt_dir <- system.file("extdata/txt", package = "postvocs")
sample_file <- system.file("extdata/SampleID.xlsx", package = "postvocs")

# Process all .txt files in the example folder
result <- batch_process_gcms(
  txt_dir = txt_dir,
  sample_file = sample_file,
  sheet = 1
)

# Access a specific sample's PeakTable
peak_sample <- result$PeakTables[[1]]

# View summary of processing
result$summary

# If you want debug output or different encoding, you can pass them:
result <- batch_process_gcms(txt_dir, sample_file, debug = TRUE)
result <- batch_process_gcms(txt_dir, sample_file, encoding = "GBK")
```

---

build_cas_abundance *Build a CAS × Sample abundance matrix from extracted peak areas*

Description

Takes a list of extracted CAS–area vectors (as returned by extract_peak_areas) and constructs a matrix where rows are CAS compounds and columns are samples. This is the standard abundance matrix used for subsequent annotation and screening.

Usage

```
build_cas_abundance(area_list)
```

Arguments

area_list	A named list of numeric vectors, each vector representing one sample with CAS numbers as names and total peak areas as values. Typically the output of extract_peak_areas().

Details

The function collects all unique CAS numbers from all samples, sorts them, and creates a matrix with CAS as rows and samples as columns. If a sample does not contain a particular CAS, its area is set to 0.

Value

A data frame with the first column "CAS" and subsequent columns for each sample (in the order they appear in area_list). Missing values are filled with 0.

Examples

```
# Get paths to example data
txt_dir <- system.file("extdata/txt", package = "postvocs")
sample_file <- system.file("extdata/SampleID.xlsx", package = "postvocs")

# Batch process example files and extract peak areas
batch <- batch_process_gcms(txt_dir, sample_file)
areas <- extract_peak_areas(batch)

# Build abundance matrix (CAS as rows, samples as columns)
abund <- build_cas_abundance(areas)

# View first few rows and columns
head(abund[, 1:5])
```

---

extract_peak_areas *Extract CAS and total peak areas from GC-MS results*

Description

This function extracts CAS numbers and their corresponding total peak areas from GC-MS PeakTable and SearchResults data. It supports three input types:

- A list from process_gcms_txt() (single sample)
- A list from batch_process_gcms() (multiple samples)
- A folder path containing PeakTable and SearchResults files (CSV or Excel)

Usage

```
extract_peak_areas(x, ...)
```

Arguments

x	Either: • A list from `process_gcms_txt()` containing `PeakTable` and `SearchResults` • A list from `batch_process_gcms()` containing `PeakTables` and `SearchResults` • A character path to a folder containing PeakTable and SearchResults files
...	Additional arguments passed to `read.csv` or `readxl::read_excel` (e.g., `sheet = 1` for Excel files).

Details

The function automatically detects the input type:

- If `x` is a character path to an existing folder, it scans for files containing "PeakTable" and "SearchResults" in their names, pairs them by common prefix, and processes all samples.
- If `x` is a list containing `PeakTable` and `SearchResults`, it processes as a single sample and returns a list with one element.
- If `x` is a list containing `PeakTables` and `SearchResults`, it processes all samples in batch.

Value

A named list. Each element is a named numeric vector with CAS as names and total peak area as values. The list names are sample names.

Examples

```
# Get paths to example data
txt_file <- system.file("extdata/txt/sample1.txt", package = "postvocs")
txt_dir <- system.file("extdata/txt", package = "postvocs")
sample_file <- system.file("extdata/SampleID.xlsx", package = "postvocs")

# 1. From process_gcms_txt result (single sample)
res <- process_gcms_txt(txt_file)
result_single <- extract_peak_areas(res)
# result_single is a list with one element named "sample1"

# 2. From batch_process_gcms result (multiple samples)
batch <- batch_process_gcms(txt_dir, sample_file)
results_batch <- extract_peak_areas(batch)
# results_batch is a list with elements named by mapped sample names
# e.g., "Site1_TreatA_1", "Site1_TreatA_2", ...

# 3. From a folder containing PeakTable and SearchResults CSV files
# (This requires you to have saved the results from save_gcms_results first)
# For example:
save_gcms_results(res, tempdir(), format = "csv")
```

```
results_folder <- extract_peak_areas(tempdir())
```

`filter_by_frequency` *Filter compounds by occurrence frequency across samples and treatment groups*

Description

This function performs a two-step filtering of compounds from an abundance matrix. It optionally removes compounds detected in blank samples, then retains compounds based on occurrence frequency across all samples and within treatment groups. It returns a detailed summary table with flags for each compound.

Usage

```
filter_by_frequency(
  abundance_data,
  sample_group_file,
  sheet = 1,
  group_col = "Combined_Treatment",
  threshold_total = 0.25,
  threshold_treatment = 0.25,
  blank_indicators = c("Species", "Treatment", "Combined_Treatment"),
  remove_blank_compounds = FALSE
)
```

Arguments

`abundance_data`	Either a data.frame with a first column named "CAS" and a second column named "Compound_Name", a character path to a CSV or Excel file with the same structure, or a list returned by `annotate_compounds()` (containing `abundance_updated`).
`sample_group_file`	Either a data.frame or a character path to the sample grouping file (CSV or Excel). The first column must be "FileID" (ignored) and the second column must be "SampleName" (used to match abundance columns).
`sheet`	Integer or character. Sheet number or name to read from Excel ‘sample_group_file‘. Default is ‘1‘ (first sheet). Ignored if ‘sample_group_file‘ is a data.frame or CSV file.
`group_col`	Character. Name of the column in ‘sample_group_file‘ that defines treatment groups. Default is ‘"Combined_Treatment"‘.
`threshold_total`	Numeric. Minimum fraction of total samples in which a compound must appear to pass the first filter. Default is ‘0.25‘ (25 percent).
`threshold_treatment`	Numeric. Minimum fraction of samples within a treatment group for the second filter. Default is ‘0.25‘ (25 percent).

`blank_indicators`
: Character vector. Columns in ‘sample_group_file‘ that must all equal "0" to identify blank samples. Default is ‘c("Species", "Treatment", "Combined_Treatment")‘. If some columns are not present, they are skipped with a warning.

`remove_blank_compounds`
: Logical. If ‘TRUE‘, compounds detected (area > 0) in any blank sample are removed from all samples. If ‘FALSE‘ (default), blank samples are still excluded from frequency calculations, but compounds detected in blanks are retained in the abundance matrix.

Value

A list with five components:

summary A data frame with filtering statistics.

abundance The full abundance matrix with added frequency columns and a flag column indicating whether each compound was detected in blank samples.

round1 Data frame of compounds kept in round 1.

round2 Data frame of compounds kept in round 2.

summary_table A comprehensive table for all original compounds with flags.

`process_gcms_txt` *Parse a single GC-MS exported TXT file*

Description

Extracts the [MC Peak Table] and [MS Similarity Search Results for Spectrum Process Table] from a Shimadzu GC-MS workstation text export. The function returns the two tables as data frames in a list, with element names derived from the input file name. No files are written to disk.

Usage

```
process_gcms_txt(txt_file, debug = FALSE, encoding = "UTF-8")
```

Arguments

`txt_file`
: Character. Path to a single GC-MS exported TXT file (e.g., `"data-raw/03.qgd.txt"`). The file name (without extension) is used to name the output list elements.

`debug`
: Logical. If `TRUE`, prints debugging information such as total lines, start positions, and column names. Default is `FALSE`.

`encoding`
: Character. Encoding of the input TXT file. Default is `"UTF-8"`. Change to `"GBK"` or `"latin1"` if the file contains non-UTF-8 characters (e.g., in NIST library paths).

Details

The function expects the TXT file to have the following structure:

- A header line starting with "Data File Name" (used only for debug and not for naming).
- After [MC Peak Table], a line like # of Peaks\t38 or # of Peaks,38 giving the number of peaks.
- The column name line of the peak table must start with "Peak#" (e.g., Peak#\tRet.Time\tArea... or Peak#,Ret.Time,Area...).
- The mass spectral search results section starts with [MS Similarity Search Results ...], and its column name line must start with "Spectrum#".

The delimiter (tab or comma) is auto-detected from the column name line of each table and used for parsing all rows of that table. The two tables may use different delimiters.

Value

A list with two data frames, named as "<basename>_PeakTable" and "<basename>_SearchResults", where <basename> is the input file name without extension.

- **PeakTable**: The extracted peak table from [MC Peak Table], containing columns like Peak#, Ret.Time, Area, etc.
- **SearchResults**: The extracted similarity search results from [MS Similarity Search Results ...], containing columns like Spectrum#, CAS#, Name, etc. CAS numbers are cleaned by removing all spaces.

Examples

```
# Get path to the example data directory
txt_dir <- system.file("extdata/txt", package = "postvocs")

# Construct the full path to a sample TXT file
txt_file <- file.path(txt_dir, "sample1.txt")

# Process the file (default encoding UTF-8)
res <- process_gcms_txt(txt_file, debug = TRUE)

# If encountering encoding warnings, try GBK
res <- process_gcms_txt(txt_file, encoding = "GBK")

# Access the two tables using dynamically generated names
peak_table <- res[["sample1_PeakTable"]]
search_results <- res[["sample1_SearchResults"]]

# See all names
names(res)
```

save_gcms_results *Save GC-MS parsing results to CSV or Excel files*

Description

Saves the data frames from process_gcms_txt() or batch_process_gcms() to individual files. For SearchResults data frames, the CAS column is prefixed with a single quote to prevent Excel from interpreting CAS numbers as dates.

Usage

```
save_gcms_results(results, output_dir, format = c("csv", "xlsx"), ...)
```

Arguments

results	A list returned by process_gcms_txt() or batch_process_gcms(). For process_gcms_txt, the list contains two data frames (PeakTable and SearchResults). For batch_process_gcms, the list contains components PeakTables and SearchResults, each a named list of data frames.
output_dir	Character. Path to the directory where files will be saved. Created if it does not exist.
format	Character. Output format: "csv" or "xlsx". Default is "csv".
...	Additional arguments passed to write.csv (if format = "csv") or openxlsx::writeData (if format = "xlsx"). For CSV, common arguments include row.names = FALSE.

Details

The function detects data frames that contain CAS-related columns (based on the data frame name containing "SearchResults") and adds a single quote prefix to the CAS column to avoid Excel date conversion. For PeakTable data frames, no modification is applied.

If format = "xlsx", the openxlsx package is required. If not installed, the function will prompt to install it.

Value

Invisibly, a character vector of saved file paths.

Examples

```
# Get paths to example data
txt_dir <- system.file("extdata/txt", package = "postvocs")
txt_file <- file.path(txt_dir, "sample1.txt")
sample_file <- system.file("extdata/SampleID.xlsx", package = "postvocs")

# (1) Process a single file and save as CSV
res <- process_gcms_txt(txt_file)
save_gcms_results(res, output_dir = tempdir(), format = "csv")
```

```
# (2) Batch process and save as Excel (all in one file)
batch <- batch_process_gcms(txt_dir, sample_file)
save_gcms_results(batch, output_dir = tempdir(), format = "xlsx")

# (3) Save only PeakTables (custom extraction)
save_gcms_results(batch$PeakTables, output_dir = tempdir(), format = "csv")
```

`save_postvocs_results` *Save postvocs results to CSV or Excel files*

Description

This function saves the results from ‘build_cas_abundance()‘, ‘annotate_compounds()‘, or ‘filter_by_frequency()‘ to CSV or Excel files. It automatically adds a single quote prefix to CAS columns to prevent Excel date conversion. It checks whether the CAS column already contains a leading single quote; if so, it does not add another one.

Usage

```
save_postvocs_results(
  x,
  output_dir,
  prefix = NULL,
  format = c("csv", "xlsx"),
  what = NULL,
  ...
)
```

Arguments

`x`	Either: • A data.frame (e.g., from ‘build_cas_abundance()‘) • A list from ‘annotate_compounds()‘ (contains ‘annotation‘ and ‘abundance_updated‘) • A list from ‘filter_by_frequency()‘ (contains ‘summary‘, ‘abundance‘, ‘round1‘, ‘round2‘, ‘summary_table‘)
`output_dir`	Character. Directory where files will be saved. Created if it does not exist. **Must be specified** (no default).
`prefix`	Character. Optional prefix for output file names. If ‘NULL‘, a prefix is automatically derived from the input: • For data.frame: ‘"abundance"‘ • For annotate_compounds: ‘"annotated"‘ • For filter_by_frequency: ‘"filtered"‘
`format`	Character. Output format: ‘"csv"‘ or ‘"xlsx"‘. Default is ‘"csv"‘.

what	Character vector. Which components to save when ‘x‘ is a list. For ‘annotate_compounds‘ results, can be ‘"annotation"‘ and/or ‘"abundance_updated"‘ (default both). For ‘filter_by_frequency‘ results, can be any of ‘"summary"‘, ‘"abundance"‘, ‘"round1"‘, ‘"round2"‘, ‘"summary_table"‘ (default all except ‘summary‘ because it’s just a small stats table). Set to ‘"all"‘ to save all components.
...	Additional arguments passed to ‘write.csv‘ (if ‘format = "csv"‘) or ‘openxlsx::writeData‘ (if ‘format = "xlsx"‘). Common arguments include ‘row.names = FALSE‘.

Details

For Excel output (‘format = "xlsx"‘), if the input is a list containing multiple data frames (e.g., from ‘filter_by_frequency()‘ or ‘annotate_compounds()‘), all tables are saved as separate worksheets in a single Excel file. For CSV output, each table is saved as an independent file.

For data.frames that contain a column named "CAS", the function adds a single quote prefix to each CAS value to prevent Excel from interpreting them as dates. It does not add a second quote if the value already starts with a single quote.

If ‘format = "xlsx"‘, the ‘openxlsx‘ package is required.

Value

Invisibly, a character vector of saved file paths.

Examples

```
# Load pre-annotated example data
annotated <- readRDS(system.file("extdata/annotated_example.rds", package = "postvocs"))

# Path to sample grouping file
sample_file <- system.file("extdata/SampleID.xlsx", package = "postvocs")

# Perform frequency-based screening (if needed)
result <- filter_by_frequency(
  abundance_data = annotated,
  sample_group_file = sample_file,
  group_col = "Combined_Factor",
  blank_indicators = c("Factor1", "Factor2", "Combined_Factor")
)

# Save results
save_postvocs_results(result, output_dir = tempdir(), format = "csv")
```

Index

∗ **postvocs**
annotate_compounds, 2
batch_process_gcms, 4
build_cas_abundance, 5
extract_peak_areas, 6
filter_by_frequency, 8
process_gcms_txt, 9
save_gcms_results, 11
save_postvocs_results, 12

annotate_compounds, 2

batch_process_gcms, 4
build_cas_abundance, 5

extract_peak_areas, 6

filter_by_frequency, 8

process_gcms_txt, 9

save_gcms_results, 11
save_postvocs_results, 12