Package 'pepVet'

September 18, 2026

Type Package

Title Evaluate Proteolytic Digests for Proteomics Workflows

Version 0.99.1

Description Simulates proteolytic digestion, scores the resulting peptides for LC-MS/MS suitability, compares candidate enzymes, and reports digest quality at the protein level. Supports 40 cleaver-compatible enzyme rules, workflow presets, peptide mass and pI calculations, sequence-local cleavage-efficiency annotations, and proteome-aware uniqueness scoring. Evaluates multi-FASTA files in batches with per-protein triage and proteome-level summaries. Exports peptide lists for Skyline and generic downstream tools and prints styled console reports.

License MIT + file LICENSE

URL https://github.com/LangeLab/pepVet, https://langelab.github.io/pepVet/

BugReports https://github.com/LangeLab/pepVet/issues

Depends R (>= 4.6.0)

Imports Biostrings, cleaver, cli, IRanges, rlang, tibble

Suggests BiocStyle, ggplot2, patchwork, ragg, testthat (>= 3.0.0), withr, knitr, rmarkdown

VignetteBuilder knitr

biocViews Proteomics, MassSpectrometry, QualityControl

Encoding UTF-8

Config/Needs/build BiocStyle, ggplot2, knitr, patchwork, rmarkdown

Config/Needs/check rcmdcheck

Config/Needs/coverage covr, ggplot2, patchwork, ragg, testthat (>= 3.0.0), withr, xml2

Config/Needs/lint lintr

Config/Needs/website BiocStyle, ggplot2, knitr, patchwork, pkgdown, ragg, rmarkdown

Config/testthat/edition 3

Roxygen list(markdown = TRUE)

Config/roxygen2/version 8.0.0

git_url https://git.bioconductor.org/packages/pepVet

git_branch devel

git_last_commit 6582caf

git_last_commit_date 2026-07-30

Repository Bioconductor 3.24

Date/Publication 2026-09-17

Author Enes K. Ergin [aut, cre] (ORCID:
<https://orcid.org/0000-0001-9810-7399>)

Maintainer Enes K. Ergin <eneskemalergin@gmail.com>

Contents

`pepVet-package` *pepVet: Evaluate Proteolytic Digests for Proteomics Workflows*

Description

Simulates proteolytic digestion, scores theoretical digest products for pre-acquisition review, compares candidate enzymes, and reports digest quality at the protein level. Includes 40 cleaver-compatible enzyme rules, workflow presets, peptide mass and pI utilities, batch evaluation, and compact console reporting.

Author(s)

Maintainer: Enes K. Ergin <eneskemalergin@gmail.com> (ORCID)

Authors:

- Enes K. Ergin <eneskemalergin@gmail.com> (ORCID)

See Also

Useful links:

- https://github.com/LangeLab/pepVet
- https://langelab.github.io/pepVet/
- Report bugs at https://github.com/LangeLab/pepVet/issues

`aa_properties` *Amino acid properties*

Description

Curates a reference table for the 22 amino acids (20 standard + U + O) used by pepVet scoring and validation utilities. The `molecular_weight` column stores free amino acid monoisotopic masses rather than peptide residue masses. Subtract the mass of water (18.01056 Da) to get residue-level masses. The `pKa_side_chain` column stores conventional reference values for the ionizable side chains C, D, E, H, K, R, Y, and U. Non-ionizable residues are recorded as `NA`.

Usage

```
aa_properties
```

Format

A tibble with 22 rows and 6 variables:

amino_acid Single-letter amino acid code.

molecular_weight Free amino acid monoisotopic mass in daltons.

residue_monoisotopic_mass Residue monoisotopic mass in daltons, equal to `molecular_weight - 18.01056`.

hydrophobicity Kyte-Doolittle hydrophobicity value.

pKa_side_chain Conventional side-chain reference pKa, or `NA` for non-ionizable residues.

is_basic Logical flag for basic residues H, K, and R.

Value

A tibble with 22 rows and six columns; see Format for column definitions.

Source

Hydrophobicity values follow Kyte J, Doolittle RF (1982). "A simple method for displaying the hydropathic character of a protein." *Journal of Molecular Biology*, 157(1), 105-132. doi:10.1016/00222836(82)905150.

Monoisotopic masses follow the free amino acid convention used by ExPASy FindMod: https://web.expasy.org/findmod/findmod_masses.html.

Side-chain pKa values follow conventional reference values summarized by Thurlkill RL, Grimsley GR, Scholtz JM, Pace CN (2006). "pK values of the ionizable groups of proteins." *Protein Science*, 15(5), 1214-1218. doi:10.1110/ps.051840806; and Pace CN, Grimsley GR, Scholtz JM (2009). "Protein ionizable groups: pK values and their contribution to protein stability and solubility." *Journal of Biological Chemistry*, 284(20), 13285-13289. doi:10.1074/jbc.R800080200.

`annotate_cleavage_sites` *Annotate cleavage-site efficiency*

Description

`annotate_cleavage_sites()` classifies sequence-local cleavage-site efficiency for trypsin-family digests. Companion to `digest_protein()` for inspecting cleavage hotspots before or alongside peptide generation.

Usage

```
annotate_cleavage_sites(sequence, enzyme = "trypsin")
```

Arguments

`sequence`	Protein input supplied as a single sequence, a named character vector of length 1, a `Biostrings::AAString`, or a FASTA file path resolving to exactly one protein. `NULL`, length-zero, and wrong-type inputs raise `pepvet_error_invalid_input`; `NA` and empty strings raise `pepvet_error_invalid_sequence`. Malformed FASTA files raise a classed `pepvet_error_invalid_input` error.
`enzyme`	Cleavage rule name. Defaults to `"trypsin"`. Cleavage-efficiency annotations are implemented for the trypsin family only: `trypsin`, `trypsin-high`, `trypsin-low`, and `trypsin-simple`. Supported non-trypsin names raise `pepvet_error_unsupported_cleavage_` missing, empty, wrong-type, and unrecognized values raise `pepvet_error_invalid_enzyme`.

Details

The current annotations are sequence-local and based on P1-P1' context only. They do not model higher-order structural accessibility, extended subsite preferences beyond P1', or PTMs that block cleavage. Unsupported enzymes raise an error rather than returning a partially annotated table.

Value

A tibble with one row per candidate cleavage site and the columns `position`, `residue`, `flanking_context`, `efficiency`, and `rule_applied`. Returns an empty tibble when the sequence contains no trypsin cleavage sites.

Limitations

Trypsin family only, sequence-local P1-P1' context only, no extended subsite modeling.

See Also

Other digest: `digest_protein()`

Examples

```
annotate_cleavage_sites("AKRTPK", enzyme = "trypsin")
```

---

batch_compare_enzymes *Compare multiple enzymes across a full proteome*

Description

`batch_compare_enzymes()` scores every protein in `sequences` against each enzyme in `enzymes` and returns a tidy tibble with one row per protein-enzyme pair. The input proteome is parsed once; each enzyme then calls `batch_evaluate()` with `cores` workers that split proteins into equal chunks processed via fork copy-on-write (`parallel::mclapply`). This means `cores` workers are fully utilised regardless of how many enzymes are compared, and the speedup applies equally to single-enzyme calls.

Usage

```
batch_compare_enzymes(
  sequences,
  enzymes = c("trypsin", "lysc", "chymotrypsin-high", "asp-n endopeptidase",
    "arg-c proteinase"),
  cores = 1L,
  missed_cleavages = 1L,
  proteome = NULL,
  weights = NULL,
  ...
)
```

Arguments

`sequences`	Multi-protein input. Accepts the same forms as `digest_protein()`. Must resolve to at least one protein. If `NULL` or empty, raises an error.
`enzymes`	Character vector of unique enzyme names to compare. Each name must be one of pepVet's supported enzyme names. Defaults to a panel of five commonly compared enzymes: trypsin, lysc, chymotrypsin-high, asp-n endopeptidase, and arg-c proteinase.

`cores` Number of parallel workers passed to `batch_evaluate()` for each enzyme. Proteins are split into `cores` equal chunks and processed with `parallel::mclapply()` on Unix or `parallel::parLapply()` on Windows. Enzymes are always processed sequentially.

`missed_cleavages` Maximum missed cleavages passed to `batch_evaluate()` for every enzyme. Defaults to `1L`.

`proteome` Optional proteome digest tibble passed to `batch_evaluate()` for every enzyme. When `NULL` (default), no uniqueness scoring and the `S_unique` column is omitted.

`weights` Optional scoring weight vector passed to `batch_evaluate()`. When `NULL` (default), uses pepVet's default scoring weights.

`...` Additional scoring arguments passed to `batch_evaluate()`, such as `gravy_range` and `length_range`.

Value

A tibble of class `pepvet_batch_comparison` with one row per protein-enzyme pair. Columns: `protein_id`, `enzyme` (factor ordered by the input `enzymes` vector, so ggplot2 axis and facet order matches your specification), then all columns returned by `batch_evaluate()`: `protein_length`, `n_peptides`, `n_valid_peptides`, component scores, `composite_score`, `verdict`, `median_peptide_length`, and the four difficulty flags. Printing shows a per-enzyme summary table before the tibble rows.

Limitations

Enzymes run sequentially even when `cores > 1`; only proteins within each enzyme are parallelized. Windows socket workers serialize the input for each enzyme, so their memory and startup costs differ from Unix fork workers.

See Also

`batch_evaluate()`, `compare_digests()`, `summarize_batch()`

Other evaluation: `batch_evaluate()`, `compare_digests()`, `evaluate_digest()`, `recommend_enzyme()`, `sensitivity_analysis()`, `summarize_batch()`, `triage_proteins()`

Examples

```
small <- system.file(
  "extdata", "small_proteome_50_proteins.fasta",
  package = "pepVet"
)
result <- batch_compare_enzymes(small, enzymes = c("trypsin", "lysc"))
result
result[result$enzyme == "trypsin", c("protein_id", "composite_score")]
```

batch_evaluate *Batch-evaluate multiple proteins*

Description

batch_evaluate() processes proteins in bulk, using one digest and score call per serial or parallel chunk, and returns a flat tibble with one row per protein. Columns include protein_id, protein_length, and all component scores, including composite_score, verdict, n_peptides, n_valid_peptides, and median_peptide_length, plus four sequence-level difficulty flags. Pass the result to summarize_batch() for aggregate statistics or to triage_proteins() for action labels.

Usage

```
batch_evaluate(
  sequences,
  enzyme = "trypsin",
  missed_cleavages = 1L,
  include_cleavage_efficiency = FALSE,
  proteome = NULL,
  weights = NULL,
  cores = 1L,
  ...
)
```

Arguments

sequences	Multi-protein input. Accepts the same forms as digest_protein(). Must resolve to at least one protein. If NULL or empty, raises an error.
enzyme	Enzyme name passed to digest_protein(). Defaults to "trypsin".
missed_cleavages	Maximum missed cleavages. Defaults to 1L.
include_cleavage_efficiency	Logical flag passed to digest_protein(). Defaults to FALSE. When TRUE, each per-protein peptide table includes a cleavage_efficiency column (does not affect the flat batch tibble columns).
proteome	Optional proteome digest tibble passed to score_peptides() for every protein evaluation. When NULL (default), no uniqueness scoring is performed and the S_unique column is omitted.
weights	Optional scoring weight vector passed to score_peptides(). When NULL (default), uses pepVet's default scoring weights.
cores	Number of parallel workers for protein-level chunking. 1L (default) runs sequentially with no extra dependencies. Values greater than 1L split the input into equal chunks and process each chunk with parallel::mclapply() on Unix or a parallel::parLapply() socket cluster on Windows.
...	Additional scoring arguments passed to score_peptides(), such as gravy_range and length_range.

Details

The returned tibble carries a `scoring_config` attribute containing the resolved ranges, weights, proteome-aware mode, and pI mode. Batch inputs must have unique protein identifiers. Difficulty flags use the active scoring ranges; `flag_short_protein` and `flag_low_complexity` remain sequence-level heuristics. Rows follow the supplied input-record order. Reordering uniquely named records changes row order but not the named per-protein results.

Value

A tibble with one row per protein. Fixed columns: `protein_id`, `protein_length`, `n_peptides`, `n_valid_peptides`, all available component scores (`S_length`, `S_coverage`, `S_count`, `S_hydro`, `S_charge`; `S_unique` when proteome is provided), `composite_score`, `verdict`, `median_peptide_length`, `flag_short_protein`, `flag_hydrophobic`, `flag_low_complexity`, `flag_no_valid_peptides`.

Limitations

Windows socket workers receive serialized copies of their input, while Unix fork workers use copy-on-write memory. If a worker or socket cluster fails, the affected chunks are retried sequentially with a warning.

See Also

`evaluate_digest()`, `compare_digests()`, `summarize_batch()`, `triage_proteins()`

Other evaluation: `batch_compare_enzymes()`, `compare_digests()`, `evaluate_digest()`, `recommend_enzyme()`, `sensitivity_analysis()`, `summarize_batch()`, `triage_proteins()`

Examples

```
small_proteome <- system.file(
  "extdata", "small_proteome_50_proteins.fasta",
  package = "pepVet"
)
batch <- batch_evaluate(small_proteome, enzyme = "trypsin")
nrow(batch)
batch[, c("protein_id", "composite_score", "verdict")]
```

---

calculate_peptide_mass

Calculate peptide mass or m/z

Description

`calculate_peptide_mass()` computes the neutral monoisotopic mass of one or more unmodified peptide sequences using residue masses stored in aa_properties. When `charge > 0`, the function returns monoisotopic m/z values using the proton mass `1.007276` Da.

Usage

```
calculate_peptide_mass(sequence, charge = 0L)
```

Arguments

sequence	Peptide sequence supplied as a character vector. Amino-acid codes must be one-letter uppercase or lowercase symbols from the 22 supported residues in aa_properties. If NULL, raises an error.
charge	Optional non-negative integer scalar or integer vector with length matching sequence. Defaults to 0L. 0L returns neutral masses. Positive values return m/z. If NULL, raises an error.

Details

This function computes masses for unmodified peptide sequences only. It does not account for chemical labels such as TMT or iTRAQ, isotopic labels such as SILAC, or post-translational modifications.

Value

A numeric vector of neutral masses or m/z values. Names are preserved from sequence when present.

Limitations

This function computes monoisotopic mass only. Average mass is not supported.

See Also

Other utils: calculate_pI(), pepvet_preset()

Examples

```
calculate_peptide_mass("PEPTIDE")
calculate_peptide_mass(c(a = "PEPTIDE", b = "AAAAAAAR"), charge = 2L)
```

---

calculate_pI *Calculate peptide isoelectric point*

Description

calculate_pI() estimates peptide isoelectric points with a bisection search over the Henderson-Hasselbalch net-charge equation. The calculation uses hard-coded terminal pKa values of 8.0 for the N-terminus and 3.1 for the C-terminus, plus side-chain pKa values from aa_properties for C, D, E, H, K, R, Y, and U.

Usage

```
calculate_pI(sequence)
```

Arguments

sequence	Peptide sequence supplied as a character vector. Amino-acid codes must be one-letter uppercase or lowercase symbols from the 22 supported residues in aa_properties. If NULL, raises an error.

Value

A numeric vector of estimated peptide pI values. Names are preserved from `sequence` when present.

Limitations

pI estimation uses the Lehninger pKa set. Calculation may be slow for more than 5000 sequences.

See Also

Other utils: `calculate_peptide_mass()`, `pepvet_preset()`

Examples

```
calculate_pI("PEPTIDE")
calculate_pI(c("AAAAAAAR", "ACDEFGHIKLMNPQRSTVWY"))
```

---

compare_digests *Compare multiple enzymes on a single protein*

Description

`compare_digests()` runs `evaluate_digest()` for each enzyme in `enzymes` and returns a tibble of scores sorted by `composite_score` descending. Main model-ranking function for pre-acquisition enzyme review.

Usage

```
compare_digests(
  sequence,
  enzymes = c("trypsin", "lysc"),
  missed_cleavages = 1L,
  proteome = NULL,
  weights = NULL,
  ...
)
```

Arguments

`sequence`	A single-protein input. Accepts the same forms as `digest_protein()` but must resolve to exactly one protein. If `NULL` or empty, raises an error.
`enzymes`	Character vector of unique enzyme names to compare. Defaults to `c("trypsin", "lysc")`. Each name must be one of pepVet's supported cleaver-compatible enzyme names.
`missed_cleavages`	Maximum missed cleavages passed to `digest_protein()` for every enzyme. Defaults to `1L`.
`proteome`	Optional proteome digest tibble passed to `score_peptides()` for all enzyme evaluations. When `NULL` (default), no uniqueness scoring.
`weights`	Optional scoring weight vector passed to `score_peptides()`. When `NULL` (default), uses pepVet's default scoring weights.

... Additional arguments passed to evaluate_digest(). This includes scoring arguments such as gravy_range, length_range, and include_pI, plus include_cleavage_efficien when peptide-level cleavage annotations are requested during comparison.

Value

A tibble with one row per enzyme and columns enzyme followed by the score columns returned by evaluate_digest(), sorted by composite_score descending.

Limitations

Single-protein only. Use batch_compare_enzymes() for proteome-wide enzyme comparison.

See Also

evaluate_digest(), recommend_enzyme()

Other evaluation: batch_compare_enzymes(), batch_evaluate(), evaluate_digest(), recommend_enzyme(), sensitivity_analysis(), summarize_batch(), triage_proteins()

Examples

```
bsa_path <- system.file("extdata", "P02769.fasta", package = "pepVet")
compare_digests(bsa_path, enzymes = c("trypsin", "lysc"))
```

---

digest_protein *Simulate a proteolytic digest*

Description

digest_protein() normalizes protein-like input, validates the amino-acid alphabet, applies a cleaver-compatible enzyme rule, and returns peptide coordinates in a tidy tibble. Entry point for all higher-level pepVet workflows: scoring, enzyme comparison, and batch evaluation.

Usage

```
digest_protein(
  sequence,
  enzyme = "trypsin",
  missed_cleavages = 1L,
  include_cleavage_efficiency = FALSE
)
```

Arguments

sequence Protein input supplied as a single sequence, a named character vector of sequences, a Biostrings::AAString, a Biostrings::AAStringSet, or a path to a FASTA file. File extension is not used to detect FASTA input; any existing file that Biostrings::readAAStringSet() can parse as FASTA is accepted. NULL, length-zero, and wrong-type inputs raise pepvet_error_invalid_input; NA or an empty string raises pepvet_error_invalid_sequence. Malformed FASTA files raise a classed pepvet_error_invalid_input error. Multi-record inputs must have unique protein identifiers; duplicates raise pepvet_error_invalid_input.

`enzyme` Cleavage rule name. Defaults to `"trypsin"`. pepVet validates this against its hard-coded registry of cleaver-compatible enzyme names, including `trypsin`, `lysc`, `glutamyl endopeptidase`, `asp-n endopeptidase`, `chymotrypsin-high`, and `thermolysin`. Missing, empty, wrong-type, and unsupported values raise `pepvet_error_invalid_enzyme`.

`missed_cleavages` Maximum number of missed cleavages to include. Must be a single non-negative integer. Defaults to `1L`. Invalid values raise `pepvet_error_invalid_missed_cleavages`.

`include_cleavage_efficiency` Logical flag indicating whether to append a `cleavage_efficiency` column to the peptide output. Defaults to `FALSE`. Trypsin-family digests receive sequence-local high/medium/low annotations; unsupported enzymes return `NA` in this optional column. The flag must be a single non-missing logical value; invalid values raise `pepvet_error_invalid_include_cleavage_efficiency`.

Details

FASTA record names are preserved as `protein_id` values when they are present, including irregular headers that do not use UniProt pipe formatting. Unnamed input sequences receive generated `sequence_<n>` IDs. Multi-record identifiers must remain unique after unnamed records receive generated IDs. Protein groups follow the supplied record order. Reordering records changes group order only; each record retains the same peptide values and coordinates. pepVet uses cleaver-compatible cleavage rules for the strict cut sites and expands missed cleavages itself so repeated peptides and overlapping ranges retain exact start and end coordinates. Peptides are returned whether or not they later count as valid in the pepVet scoring model. Validity is a separate decision controlled by `score_peptides()` through `length_range`.

When cleavage-efficiency annotations are requested, pepVet records the weakest annotated efficiency class across cleavage sites that delimit or fall within a peptide. These annotations reflect local P1-P1' sequence context only. They do not model extended subsite preferences, structural protection, or PTMs that alter cleavage behavior.

Value

A tibble with one row per peptide and the columns `protein_id`, `peptide`, `start`, `end`, `length`, and `missed_cleavages`. Each row represents one theoretical cleavage product for one protein under the selected enzyme rule and missed-cleavage allowance. When `include_cleavage_efficiency = TRUE`, the output also includes a `cleavage_efficiency` column. The `cleavage_efficiency` values are all `NA` for enzymes outside the trypsin family.

Limitations

Cleaver rules only, no structural accessibility, no PTM awareness.

See Also

Other digest: `annotate_cleavage_sites()`

Examples

```
digest_protein("MKWVTFISLLFLFSSAYSR")
digest_protein(Biostrings::AAString("MKWVTFISLLFLFSSAYSR"))
digest_protein("AKRTPK", include_cleavage_efficiency = TRUE)
```

digest_report *Print a compact console report for a proteolytic digest*

Description

digest_report() formats the output of evaluate_digest() or compare_digests() as a human-readable console summary. Reports use ASCII-safe labels and score profiles, wrap long protein identifiers, and simplify comparison tables in narrow terminals. The function returns its input invisibly so it can be used in pipelines. Use it when you want a compact review of component scores during interactive enzyme selection or package-level demonstrations.

Usage

```
digest_report(x, title = NULL)
```

Arguments

x	The object to report on. Accepts: **Named list from** evaluate_digest() Prints a single-protein component-score summary for the evaluated enzyme. **Tibble from** compare_digests() Prints a multi-enzyme ranking table with the highest-scoring enzyme highlighted. If NULL or an unrecognised type, raises an error.
title	Optional character string printed as a section header above the report. When NULL (default), a report-type heading is used. A non-NULL value must be a single character string.

Value

x, invisibly.

Limitations

Output is printed to the console only. File output is not supported.

See Also

evaluate_digest(), compare_digests()

Other report: pepvet_check()

Examples

```
bsa_path <- system.file("extdata", "P02769.fasta", package = "pepVet")
ev <- evaluate_digest(bsa_path)
digest_report(ev)

comp <- compare_digests(bsa_path, enzymes = c("trypsin", "lysc"))
digest_report(comp)
```

`evaluate_digest` *Evaluate a proteolytic digest*

Description

`evaluate_digest()` combines `digest_protein()` and `score_peptides()` into a single call and returns a named list containing the peptide table, the score table, and the resolved input parameters. Use it when you want a full digest object for one protein and one enzyme without manually wiring the two lower-level functions together.

Usage

```
evaluate_digest(
  sequence,
  enzyme = "trypsin",
  missed_cleavages = 1L,
  include_cleavage_efficiency = FALSE,
  proteome = NULL,
  weights = NULL,
  ...
)
```

Arguments

`sequence`	Protein input. Accepts the same forms as `digest_protein()`: a character sequence, named character vector, `Biostrings::AAString`, `Biostrings::AAStringSet`, or a FASTA file path. If `NULL` or empty, raises an error. Multi-record inputs must have unique protein identifiers.
`enzyme`	Enzyme name passed to `digest_protein()`. Defaults to `"trypsin"`.
`missed_cleavages`	Maximum missed cleavages passed to `digest_protein()`. Defaults to `1L`.
`include_cleavage_efficiency`	Logical flag passed to `digest_protein()`. When `TRUE`, the returned peptide table gains a `cleavage_efficiency` column. This does not affect the score components.
`proteome`	Optional proteome digest tibble passed to `score_peptides()` for peptide uniqueness scoring. When `NULL` (default), no uniqueness scoring is performed.
`weights`	Optional scoring weight vector passed to `score_peptides()`. When `NULL` (default), uses pepVet's default scoring weights. When scoring a non-tryptic digest directly, `evaluate_digest()` forwards the selected `enzyme` so enzyme-aware S_count denominators stay aligned with the digest.
`...`	Additional scoring arguments passed to `score_peptides()`, such as `gravy_range` and `length_range`. This makes workflow presets from `pepvet_preset()` directly compatible with `evaluate_digest()` through `do.call()` or argument splicing.

Details

`evaluate_digest()` preserves pepVet's scoring metadata so the returned object can be interpreted honestly outside the immediate scoring call. In particular, `params$preset_used` records whether the resolved scoring configuration matches one of pepVet's named presets or should be treated as `"custom"`. `params` also stores the resolved GRAVY and length ranges, active weights, proteome-aware mode, and pI mode. The cleavage-efficiency counts summarize annotated trypsin-family cleavage sites only; unsupported enzymes receive `NA` in these informational fields. For uniquely named multi-record input, score rows, peptide groups, and `params$protein_ids` follow the supplied record order. Reordering input records changes that presentation order but not the named per-protein results.

Value

A named list with three elements:

`scores` A tibble from `score_peptides()` with one row per protein, plus the informational columns `n_high_efficiency_sites` and `n_low_efficiency_sites`.

`peptides` A tibble from `digest_protein()` with one row per peptide.

`params` A list recording the resolved `enzyme` name, `missed_cleavages` count, `protein_ids` found in the input, the resolved `preset_used` label, GRAVY and length ranges, active weights, proteome-aware mode, and pI mode.

Limitations

Cleavage efficiency annotations only work for trypsin-family enzymes. Unsupported enzymes receive `NA` counts. The scoring model is rule-based; see `score_peptides()` for scope details.

See Also

`digest_protein()`, `score_peptides()`, `compare_digests()`, `batch_evaluate()`

Other evaluation: `batch_compare_enzymes()`, `batch_evaluate()`, `compare_digests()`, `recommend_enzyme()`, `sensitivity_analysis()`, `summarize_batch()`, `triage_proteins()`

Examples

```
bsa_path <- system.file("extdata", "P02769.fasta", package = "pepVet")
result <- evaluate_digest(bsa_path, enzyme = "trypsin")
result$scores
result$params$enzyme
result$params$preset_used
```

`export_peptide_list` *Export a peptide list for downstream tools*

Description

`export_peptide_list()` filters a pepVet peptide tibble to valid peptides and returns or writes the result in a format compatible with downstream proteomics tools. Supported formats are `"skyline"`, `"generic"`, and `"fasta"`.

Usage

```
export_peptide_list(
  peptides,
  format = "skyline",
  charges = 2:3,
  length_range = c(7L, 25L),
  file = NULL
)
```

Arguments

peptides	A peptide tibble produced by digest_protein() or accessible via evaluate_digest()$peptides. Must contain at minimum the columns protein_id, peptide, and length. If NULL or NA, raises an error.
format	Export format. Defaults to "skyline".
	"skyline" A tibble (or CSV when file is specified) with columns Protein, Peptide Sequence, Precursor Charge, and Precursor Mz. One row per peptide per charge state. M/z values are computed via calculate_peptide_mass().
	"generic" A tibble (or CSV when file is specified) with all pepVet peptide columns plus a computed gravy column and a valid logical column marking peptides that pass length_range.
	"fasta" A character vector (or file when file is specified) of FASTA-formatted records for valid peptides only. Each record has a header of the form >protein_id|peptide_start-end. Requires start and end columns in peptides.
charges	Integer vector of precursor charge states for the "skyline" format. Defaults to 2:3. Ignored for other formats.
length_range	Integer vector of length 2 defining the inclusive valid peptide length window. Defaults to c(7L, 25L).
file	Optional file path. When NULL (default), returns the result. When specified, writes to that path and returns file invisibly. "skyline" and "generic" are written as CSV via utils::write.csv(); "fasta" is written with base::writeLines().

Details

Precursor m/z for Skyline export is computed as $(M + z \times 1.007276)/z$ where M is the neutral monoisotopic peptide mass and z is the charge state. Skyline accepts this format via File > Import > Transition List.

When no peptides pass the length_range filter, skyline format returns an empty tibble and fasta format returns an empty character vector.

Value

When file = NULL:

- For "skyline": a tibble with columns Protein, Peptide Sequence, Precursor Charge, and Precursor Mz.
- For "generic": a tibble with all original columns plus gravy, pI, and valid.
- For "fasta": a character vector of FASTA records.

When file is specified: file, invisibly.

Limitations

Only peptides passing the specified `length_range` are exported.

See Also

`digest_protein()`, `evaluate_digest()`, `calculate_peptide_mass()`

Examples

```
bsa_path <- system.file("extdata", "P02769.fasta", package = "pepVet")
peps <- digest_protein(bsa_path, enzyme = "trypsin")
export_peptide_list(peps, format = "skyline")
export_peptide_list(peps, format = "generic")
export_peptide_list(peps, format = "fasta")
```

---

pepvet_check *Quick digest check for a single protein*

Description

`pepvet_check()` is a convenience wrapper that evaluates a protein digest and immediately prints a compact console report. It is intended for interactive use and first-time exploration where a single call is more useful than manually wiring `evaluate_digest()` and `digest_report()`.

Usage

```
pepvet_check(sequence, enzyme = "trypsin", ...)
```

Arguments

`sequence`	Protein input. Accepts the same forms as `evaluate_digest()`. If `NULL`, raises an error.
`enzyme`	Enzyme name. Defaults to `"trypsin"`. If `NULL`, raises an error.
`...`	Additional arguments passed to `evaluate_digest()`, such as `missed_cleavages`, `include_cleavage_efficiency`, `weights`, `gravy_range`, and `length_range`.

Value

The `evaluate_digest()` result list, invisibly.

See Also

`evaluate_digest()`, `digest_report()`

Other report: `digest_report()`

Examples

```
bsa_path <- system.file("extdata", "P02769.fasta", package = "pepVet")
pepvet_check(bsa_path, enzyme = "trypsin")
```

pepvet_plot_config *Configure pepVet plot appearance*

Description

Override default colors, numeric parameters, and theme elements used by all pepVet plotting functions. Changes persist for the session. Call with no arguments to view the current configuration.

Usage

```
pepvet_plot_config(palette = NULL, params = NULL, theme = NULL)
```

Arguments

palette	Named list of color overrides. Names must match existing .pepvet_pal entries (e.g. list(brand = "#004488", good = "#2ECC71")). Sub-lists like verdict, component, and overlap are replaced entirely.
params	Named list of parameter overrides. Names must match existing .pepvet_params entries (e.g. list(verdict_good = 0.70, scatter_alpha = 0.90)).
theme	Named list of ggplot2 theme element overrides. Passed directly to ggplot2::theme() and applied on top of the base .pepvet_theme(). (e.g. list(legend.position = "right", plot.title = element_text(size = 14))).

Value

Invisibly returns a list with current palette, params, and theme.

See Also

Other plot-utils: pepvet_plot_config_reset(), pepvet_save_figure(), pepvet_theme_manuscript(), pepvet_theme_presentation()

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  # View current config
  pepvet_plot_config()

  # Customize brand color and Good threshold
  pepvet_plot_config(
    palette = list(brand = "#004488", good = "#2ECC71"),
    params  = list(verdict_good = 0.70)
  )

  # Reset to defaults
  pepvet_plot_config_reset()
}
```

pepvet_plot_config_reset *Reset pepVet plot configuration to defaults*

Description

Restores all colors, parameters, and theme overrides to the package defaults.

Usage

```
pepvet_plot_config_reset()
```

Value

Invisibly returns a list with current `palette`, `params`, and `theme`.

See Also

Other plot-utils: `pepvet_plot_config()`, `pepvet_save_figure()`, `pepvet_theme_manuscript()`, `pepvet_theme_presentation()`

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  pepvet_plot_config(palette = list(brand = "#004488"))
  pepvet_plot_config_reset()
}
```

pepvet_preset *Return a named scoring preset*

Description

`pepvet_preset()` returns a named list containing a GRAVY range, peptide length range, and scoring weights for a supported proteomics workflow. Presets are intended as editable starting points rather than hard rules. Their exact ranges and weights are conservative package choices, not empirically calibrated boundaries.

Usage

```
pepvet_preset(type = "standard")
```

Arguments

`type`	Preset name. Defaults to `"standard"`. Supported values are `"standard"`, `"dia"`, `"targeted"`, `"membrane"`, `"ffpe_degraded"`, and `"fractionated"`. If `NULL`, raises an error.

Value

A named list with `gravy_range`, `length_range`, `weights`, and `include_pI`. The returned object can be passed directly into `score_peptides()` or `evaluate_digest()` through `do.call()` or argument splicing.

Presets

`"standard"` : Default scoring configuration for routine DDA examples. Uses the `[7, 25]` length range, `[-1.0, 0.6]` GRAVY range, and default protein-only weights.

`"dia"` : A wider starting configuration for DIA or SWATH examples. Uses the `[7, 30]` length range, `[-1.0, 0.8]` GRAVY range, and a larger coverage weight than the standard preset.

`"targeted"` : A narrower starting configuration for SRM, PRM, or MRM examples. Uses the `[8, 20]` length range and `[-0.8, 0.4]` GRAVY range. When a background proteome digest is supplied, uniqueness receives 30 percent of the composite weight.

`"membrane"` : A wider hydrophobicity configuration for membrane-protein review. Uses the `[7, 30]` length range, extends the upper GRAVY boundary to `2.0`, and assigns 5 percent of the composite weight to `S_hydro`.

`"ffpe_degraded"` : A broader length configuration for exploratory work with degraded or FFPE-derived material. Uses the `[6, 30]` length range and assigns more weight to `S_count` than the standard preset.

`"fractionated"` : SCX / high-pH RP fractionation planning. Same scoring parameters as `"standard"` but with `include_pI = TRUE` to append peptide-level pI values for fractionation-aware analysis.

Limitations

The six presets encode package priors. Inspect the returned values and use explicit arguments when the experimental context calls for other choices.

See Also

Other utils: `calculate_pI()`, `calculate_peptide_mass()`

Examples

```
pepvet_preset("standard")
```

`pepvet_save_figure` *Save a pepVet figure with publication-ready defaults*

Description

Wraps `ggplot2::ggsave()` with pepVet's recommended defaults: auto-sizing based on whether the plot is a multi-panel patchwork or a single panel, 300 DPI, anti-aliased PNG via ragg when available, and white background. All arguments in `...` are passed to `ggplot2::ggsave()` and can override the defaults.

Usage

```
pepvet_save_figure(
  plot,
  filename = "pepvet_plot.png",
  width = NULL,
  height = NULL,
  dpi = 300,
  bg = "white",
  device = NULL,
  ...
)
```

Arguments

plot	A ggplot or patchwork object produced by any pepVet plot function.
filename	Character path for the output file. Extensions .png, .pdf, .svg, etc. are handled by ggplot2::ggsave(). Defaults to "pepvet_plot.png" in the working directory.
width, height	Finite positive numeric plot dimensions in inches. When NULL (default), auto-sized: single-panel = 10x7, multi-panel patchwork = 14x10. Invalid values raise pepvet_error_invalid_plot before a graphics device is opened.
dpi	Finite positive numeric resolution in dots per inch, or one of "screen", "print", or "retina". Defaults to 300. Invalid values raise pepvet_error_invalid_plot before a graphics device is opened.
bg	Character. Background color. Defaults to "white".
device	Device to use. When NULL (default) and the filename extension is .png, tries ragg::agg_png() for anti-aliased output, falling back to "png". For other extensions, the device is inferred from the filename.
...	Additional arguments passed to ggplot2::ggsave().

Value

The saved file path invisibly.

See Also

Other plot-utils: pepvet_plot_config(), pepvet_plot_config_reset(), pepvet_theme_manuscript(), pepvet_theme_presentation()

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE) &&
  requireNamespace("patchwork", quietly = TRUE)) {
  bsa_path <- system.file("extdata", "P02769.fasta", package = "pepVet")
  res <- evaluate_digest(bsa_path, enzyme = "trypsin")
  p <- plot_digest_profile(res)
  tmp <- tempfile(fileext = ".png")
  pepvet_save_figure(p, tmp)
}
```

pepvet_theme_manuscript *pepVet manuscript theme*

Description

A compact theme for journal figures. Smaller base text (9pt), thinner grid lines, tighter margins. Use by adding to any pepVet plot: `plot + pepvet_theme_manuscript()`.

Usage

```
pepvet_theme_manuscript()
```

Value

A ggplot2 theme object.

See Also

Other plot-utils: `pepvet_plot_config()`, `pepvet_plot_config_reset()`, `pepvet_save_figure()`, `pepvet_theme_presentation()`

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  bsa_path <- system.file("extdata", "P02769.fasta", package = "pepVet")
  res <- evaluate_digest(bsa_path, enzyme = "trypsin")
  plot_length_distribution(res) + pepvet_theme_manuscript()
}
```

pepvet_theme_presentation *pepVet presentation theme*

Description

A bold theme for talks and posters. Larger base text (14pt), thicker grid lines, wider margins. Use by adding to any pepVet plot: `plot + pepvet_theme_presentation()`.

Usage

```
pepvet_theme_presentation()
```

Value

A ggplot2 theme object.

See Also

Other plot-utils: `pepvet_plot_config()`, `pepvet_plot_config_reset()`, `pepvet_save_figure()`, `pepvet_theme_manuscript()`

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  bsa_path <- system.file("extdata", "P02769.fasta", package = "pepVet")
  res <- evaluate_digest(bsa_path, enzyme = "trypsin")
  plot_length_distribution(res) + pepvet_theme_presentation()
}
```

---

plot_batch_comparison *Multi-enzyme proteome comparison*

Description

`plot_batch_comparison()` produces a four-panel side-by-side comparison of enzyme performance across a full proteome, using output from `batch_compare_enzymes()`:

Usage

```
plot_batch_comparison(comparison, title = NULL)
```

Arguments

`comparison`	A `pepvet_batch_comparison` tibble returned by `batch_compare_enzymes()`, with columns `protein_id`, `enzyme`, `composite_score`, `verdict`, and the five component score columns. A proteome-aware comparison may also contain `S_unique`, which is included in the component heatmap. If `NULL` or empty, raises an error.
`title`	Optional character title for the combined figure. When `NULL` (default), generates an auto-title with protein and enzyme counts.

Details

- **(A) Verdict summary:** 100% stacked horizontal bars showing the Good/Moderate/Poor verdict breakdown per enzyme. Enzymes are ordered by descending Good%, and the highest model summary is marked with a star.
- **(B) Score distributions:** horizontal violin plots for each enzyme, showing the full distribution of composite scores. An IQR boxplot is overlaid on each violin. Violin fill color reflects the enzyme's median verdict.
- **(C) Component heatmap:** median component scores in an enzyme-by-component grid, filled by the verdict gradient (red to amber to green). Reveals which digest quality dimension differentiates the enzymes.
- **(D) Per-protein win rate:** bar chart showing the proportion of proteins for which each enzyme achieves the highest composite score. The starred enzyme matches the top model summary in panel A.

Value

A `patchwork` object with four panels: verdict summary bars (A), score distribution violins (B), component-score heatmap (C), and per-protein win-rate bars (D).

Limitations

At proteome sizes exceeding 2000 proteins the figure panels become dense and text labels may overlap. Consider filtering the comparison to a representative subset or increasing the output dimensions.

See Also

`batch_compare_enzymes()` for the upstream comparison step.

Other plot-batch: `plot_proteome_overview()`

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE) &&
  requireNamespace("patchwork", quietly = TRUE)) {
  small <- system.file(
    "extdata", "small_proteome_50_proteins.fasta",
    package = "pepVet"
  )
  comp <- batch_compare_enzymes(small, enzymes = c("trypsin", "lysc"))
  plot_batch_comparison(comp)
}
```

---

plot_cleavage_map *Cleavage Site Map*

Description

`plot_cleavage_map()` draws the full protein as a horizontal bar and marks every cleavage site as a vertical tick, colored by efficiency (high=green, medium=amber, low=red). Peptide fragments between consecutive cleavage sites are drawn as colored blocks, with invalid peptides dimmed. When `cleavage_sites` data is not available, sites are inferred from the peptide boundaries and all rendered as the same default color.

Usage

```
plot_cleavage_map(
  result,
  cleavage_sites = NULL,
  length_range = c(7L, 25L),
  title = NULL
)
```

Arguments

`result`	A named list returned by `evaluate_digest()`. If `NULL` or invalid, raises an error.
`cleavage_sites`	Optional data.frame from `annotate_cleavage_sites()` with columns `position`, `efficiency` (character: `"high"`, `"medium"`, `"low"`), and optionally `rule`. When `NULL` (default) sites are inferred from peptide boundaries.
`length_range`	Integer vector of length 2. Valid peptide window. Defaults to `c(7L, 25L)`.
`title`	Optional character title. Auto-generated when `NULL`.

Details

When `cleavage_sites` is not supplied, cleavage positions are inferred from the C-terminal ends of MC=0 peptides (excluding the true C-terminus of the protein). Inferred sites all render in the same default color. Pass the output of `annotate_cleavage_sites()` for efficiency-aware coloring.

Value

A `ggplot` object showing a protein bar with cleavage-site ticks colored by efficiency and peptide fragment blocks between sites.

See Also

`evaluate_digest()` for the upstream digestion step, `annotate_cleavage_sites()` for efficiency-annotated cleavage sites.

Other plot-single: `plot_coverage_map()`, `plot_digest_profile()`, `plot_peptide_overlap_map()`, `plot_weight_sensitivity()`

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  bsa_path <- system.file("extdata", "P02769.fasta", package = "pepVet")
  res <- evaluate_digest(bsa_path, enzyme = "trypsin")
  p <- plot_cleavage_map(res)
  print(p)
}
```

`plot_coverage_map` *Protein Coverage Map*

Description

`plot_coverage_map()` draws a genome-browser-style view of how proteolytic peptides map onto the full protein sequence. When the input contains multiple missed-cleavage levels (e.g. MC = 0, 1, 2), each level occupies a separate horizontal lane so the cumulative coverage gain from allowing missed cleavages is immediately visible. Cleavage-site efficiency ticks and optional domain annotations can be overlaid to add mechanistic context.

Usage

```
plot_coverage_map(
  result,
  color_by = c("validity", "length_class", "hydrophobicity"),
  length_range = c(7L, 25L),
  cleavage_sites = NULL,
  domains = NULL,
  title = NULL
)
```

Arguments

result	A named list returned by evaluate_digest(). Must describe a single protein. If NULL or invalid, raises an error.
color_by	Character string selecting the peptide color scheme.
	"validity" **(default)** Blue = valid-length peptide; gray = invalid-length peptide.
	"length_class" Three-way coloring: valid (blue), too short (amber), too long (rose-red).
	"hydrophobicity" Continuous GRAVY gradient for every peptide: brand blue at the low end, green and amber through the middle, and rose-red at the high end of the displayed GRAVY range.
length_range	Integer vector of length 2. Valid peptide window. Defaults to c(7L, 25L).
cleavage_sites	Optional data.frame from annotate_cleavage_sites(). When provided, vertical ticks below the lanes mark each cleavage site, colored by efficiency: high = green, medium = amber, low = rose-red.
domains	Optional data.frame with columns name, start, end (integer, in residue positions). Each domain is drawn as a translucent background rectangle spanning all lanes with its name labelled at the top.
title	Optional character string for the plot title. Auto-generated from the protein accession and enzyme when NULL (default).

Details

Valid peptides are drawn taller than invalid peptides within each lane, maintaining a visual hierarchy that keeps validity legible regardless of color_by. When missed-cleavage levels overlap (MC >= 1), peptides within each lane are distributed into non-overlapping tracks using a greedy interval-packing algorithm, mirroring genome-browser stacking. Gap regions (residues not covered by any valid MC = 0 peptide) are highlighted with a translucent red overlay. Peptide lengths are labelled inside bars whose span is at least 2.5 % of the protein length (ensuring legibility at typical export widths); shorter bars are left unlabelled.

The color_by = "hydrophobicity" mode calls the internal Kyte-Doolittle GRAVY calculator and requires no additional input columns.

Value

A ggplot object showing a multi-lane coverage map with peptide tracks, gap highlights, optional cleavage-site efficiency ticks, and optional domain annotations.

Limitations

The interval-packing algorithm requires consistent, non-overlapping MC=0 peptide boundaries. Incomplete or conflicting peptide coordinates may produce suboptimal track layouts or raise an error during sequence reconstruction.

See Also

evaluate_digest() for the upstream digestion step, annotate_cleavage_sites() for cleavage-site annotation.

Other plot-single: plot_cleavage_map(), plot_digest_profile(), plot_peptide_overlap_map(), plot_weight_sensitivity()

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  bsa_path <- system.file("extdata", "P02769.fasta", package = "pepVet")
  res <- evaluate_digest(bsa_path, enzyme = "trypsin", missed_cleavages = 1)
  cs <- annotate_cleavage_sites(bsa_path, enzyme = "trypsin")
  p <- plot_coverage_map(res, cleavage_sites = cs)
  print(p)
}
```

---

plot_digest_profile *Four-Panel Digest Diagnostic Plot*

Description

`plot_digest_profile()` assembles a four-panel figure for a single protein-enzyme pair from an `evaluate_digest()` result. The panels are:

Usage

```
plot_digest_profile(
  result,
  length_range = c(7L, 25L),
  gravy_range = c(-1, 0.6),
  title = NULL
)
```

Arguments

`result`	A named list returned by `evaluate_digest()`. Must describe a single protein (one unique `protein_id` in `result$peptides`). If `NULL` or invalid, raises an error.
`length_range`	Integer vector of length 2. Defines the valid peptide length window, passed to the length and coverage panels. Defaults to `c(7L, 25L)`.
`gravy_range`	Numeric vector of length 2. Defines the LC-friendly GRAVY range shaded in panel B. Defaults to `c(-1.0, 0.6)`.
`title`	Optional character string for the figure title. When `NULL` (default) a title is auto-generated from the protein accession, enzyme, and missed-cleavage count.

Details

- **(A) Length distribution:** histogram of peptide lengths with the valid window shaded. Bars are colored by length class: valid (blue), too short (amber), too long (rose).
- **(B) GRAVY distribution:** histogram of GRAVY hydrophobicity scores. The LC-friendly range is shaded and bounded by dashed lines.
- **(C) Coverage map:** protein drawn as horizontal lanes (one per missed-cleavage level) with valid-length peptides stacked via a greedy interval-packing algorithm. Uncovered regions are highlighted in red. Peptide length labels appear inside segments of 8 aa or longer.

- **(D) Component scores:** horizontal bar chart for each scoring component, colored by tier (green $\geq$ 0.65, amber 0.40-0.64, red < 0.40). The composite score is marked with a dashed vertical line.

GRAVY scores are computed internally from the peptide sequences in `result$peptides` using the Kyte-Doolittle scale. No external columns are required beyond the standard `evaluate_digest()` output.

Panel C labels peptide lengths inside segments of 8 aa or longer. For heavily digested proteins this keeps the map readable without overlap. When multiple missed-cleavage levels are present, each level occupies its own horizontal lane with peptides stacked using the same greedy interval-packing algorithm as `plot_coverage_map()`.

Value

A `patchwork` object with four panels: length distribution (A), GRAVY distribution (B), coverage map (C), and component scores (D).

See Also

`evaluate_digest()` for the upstream digestion step, `plot_enzyme_comparison()` for enzyme comparison across digests.

Other plot-single: `plot_cleavage_map()`, `plot_coverage_map()`, `plot_peptide_overlap_map()`, `plot_weight_sensitivity()`

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE) &&
  requireNamespace("patchwork", quietly = TRUE)) {
  bsa_path <- system.file("extdata", "P02769.fasta", package = "pepVet")
  res <- evaluate_digest(bsa_path, enzyme = "trypsin", missed_cleavages = 1L)
  p <- plot_digest_profile(res)
  print(p)
}
```

`plot_enzyme_comparison`

Enzyme Comparison Chart

Description

`plot_enzyme_comparison()` visualises the output of `compare_digests()` as a dual-panel chart: component score bars (Panel A) and a composite score lollipop with verdict badge (Panel B).

Usage

```
plot_enzyme_comparison(
  comparison,
  scores = c("S_coverage", "S_length", "S_count", "S_hydro", "S_charge"),
  recommend = TRUE,
  title = NULL
)
```

Arguments

comparison	A tibble returned by compare_digests(). Must contain at least the columns enzyme and composite_score, plus whichever component-score columns are requested in scores. If NULL or not a data frame, raises an error.
scores	Character vector of component-score column names to display in Panel A. Any column absent from comparison is silently dropped. Defaults to all five standard component scores. S_unique can be requested when present in a proteome-aware comparison. If NULL, raises an error.
recommend	Logical. When TRUE (default), a "Top model score" badge marks the enzyme with the highest composite score in Panel B. The badge is a model ranking, not an experimental recommendation. If NULL, raises an error.
title	Optional character string for the overall plot title. Auto-generated from the protein accession when NULL (default).

Details

Panel A shows a horizontal grouped bar chart where each enzyme occupies one row and each component score is a separate colored bar, dodged side-by-side. Reference lines at the Moderate and Good verdict thresholds divide the axis into poor / moderate / good regions. Enzymes are sorted by composite score with the highest at the top.

Panel B shows the composite score as a lollipop, color-coded by verdict tier (green >= 0.65, amber 0.40-0.64, red < 0.40). When recommend = TRUE a gold "Top model score" badge is appended next to the top-ranked enzyme.

Value

A patchwork object with two panels: component-score grouped bar chart (A) and composite-score lollipop with verdict badge (B).

See Also

compare_digests(), recommend_enzyme(), plot_digest_profile(), plot_coverage_map(), plot_batch_comparison()

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE) &&
  requireNamespace("patchwork", quietly = TRUE)) {
  bsa_path <- system.file("extdata", "P02769.fasta", package = "pepVet")
  comp <- compare_digests(bsa_path,
    enzymes = c(
      "trypsin", "lysc",
      "glutamyl endopeptidase"
    )
  )
  p <- plot_enzyme_comparison(comp)
  print(p)
}
```

`plot_gravy_landscape` *GRAVY Landscape: 2D Scatter of Peptide Length vs. Hydrophobicity*

Description

`plot_gravy_landscape()` plots each peptide as a point in the length × GRAVY physicochemical space. The selected length-and-GRAVY region is shaded, points are colour-coded by window class, and marginal density panels show the 1D distributions on each axis. Peptides outside the selected region are labelled with their sequences when their count is below `label_outliers_n`.

Usage

```
plot_gravy_landscape(
  result,
  length_range = c(7L, 25L),
  gravy_range = c(-1, 0.6),
  label_outliers_n = 15L,
  title = NULL
)
```

Arguments

`result`	A named list returned by `evaluate_digest()`, **or** a data.frame / tibble with at least `length` and `gravy` columns. When a bare data.frame lacks a `gravy` column but has a `peptide` column, GRAVY scores are computed automatically. If `NULL` or an unrecognised type, raises an error.
`length_range`	Integer vector of length 2. Valid length window. Defaults to `c(7L, 25L)`. Read from `result$params` when a full `evaluate_digest()` result is supplied.
`gravy_range`	Numeric vector of length 2. LC-friendly GRAVY window. Defaults to `c(-1.0, 0.6)`. Read from `result$params` when available.
`label_outliers_n`	Integer. Maximum number of outlier points to label with their peptide sequences. Labels are suppressed when the count exceeds this threshold. Defaults to `15L`. If `NULL`, raises an error.
`title`	Optional character string for the plot title. Auto-generated when `NULL` (default).

Details

When `result` is a named list of `evaluate_digest()` results (multi-input mode), produces a faceted scatter with one panel per result (no marginal densities). GRAVY scores are computed automatically when a bare data.frame has a `peptide` column but no `gravy` column.

Value

A `patchwork` object with a central scatter of length vs GRAVY coloured by window class, and marginal density panels on the top and right axes.

See Also

evaluate_digest() for the upstream digestion step, plot_digest_profile() for a single-protein digest summary.

Other plot-distribution: plot_length_distribution(), plot_missed_cleavage_impact(), plot_mz_distribution plot_pI_distribution()

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE) &&
    requireNamespace("patchwork", quietly = TRUE)) {
  bsa_path <- system.file("extdata", "P02769.fasta", package = "pepVet")
  res <- evaluate_digest(bsa_path, enzyme = "trypsin")
  p <- plot_gravy_landscape(res)
  print(p)
}
```

---

plot_length_distribution

Standalone Peptide Length Distribution

Description

plot_length_distribution() draws a histogram of peptide lengths colour-coded by validity class (Valid / Too short / Too long), with the valid range shaded in the package's green, per-category percentage annotations, and an optional density-curve overlay.

Usage

```
plot_length_distribution(
  result,
  length_range = c(7L, 25L),
  show_density = TRUE,
  title = NULL
)
```

Arguments

result	A named list returned by evaluate_digest(), **or** a data.frame / tibble with at least a length column (e.g. the $peptides slot of such a result). If NULL or an unrecognised type, raises an error.
length_range	Integer vector of length 2 giving the valid length window c(lo, hi). Defaults to c(7L, 25L). Ignored (and read from result$params) when a full evaluate_digest() result is supplied.
show_density	Logical. When TRUE (default) a scaled kernel-density curve is overlaid on the histogram. If NULL, treated as FALSE.
title	Optional character string for the plot title. Auto-generated when NULL (default).

Details

When result is a named list of evaluate_digest() results (multi-input mode), produces a faceted panel of length distributions with one facet per result, using per-result valid-length ranges.

Value

A ggplot object showing a histogram of peptide lengths coloured by validity class with valid-range shading and optional density overlay.

See Also

evaluate_digest() for the upstream digestion step, plot_digest_profile() for a single-protein digest summary.

Other plot-distribution: plot_gravy_landscape(), plot_missed_cleavage_impact(), plot_mz_distribution(), plot_pI_distribution()

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  bsa_path <- system.file("extdata", "P02769.fasta", package = "pepVet")
  res <- evaluate_digest(bsa_path, enzyme = "trypsin")
  p <- plot_length_distribution(res)
  print(p)
}
```

---

plot_missed_cleavage_impact

Missed Cleavage Impact Plot

Description

plot_missed_cleavage_impact() visualizes how allowing more missed cleavages changes each component score and the composite. The user runs evaluate_digest() at MC = 0, 1, 2 (or more) and passes the results as a named list. Each component score is drawn as a connected line; the composite score is drawn as a bold line. An annotation marks the MC count that maximizes the composite.

Usage

```
plot_missed_cleavage_impact(
  results,
  components = c("S_length", "S_coverage", "S_count", "S_hydro", "S_charge"),
  title = NULL
)
```

Arguments

results	A named list of evaluate_digest() results. Names should be the MC level (e.g., list("MC=0" = r0, "MC=1" = r1, "MC=2" = r2)). **Or** an unnamed list of length 2-4, in which case names are auto-assigned as "MC=0", "MC=1", etc. All results must use the same protein and enzyme; only the missed-cleavage setting may differ. If NULL, raises an error.
components	Character vector of component score columns to show. Defaults to c("S_length","S_coverage","S If NULL, raises an error.
title	Optional character title. Auto-generated when NULL.

Value

A ggplot object showing connected line plots of component and composite scores across missed-cleavage levels, with an annotation at the MC count that maximizes the composite.

See Also

evaluate_digest() for the upstream digestion step.

Other plot-distribution: plot_gravy_landscape(), plot_length_distribution(), plot_mz_distribution(), plot_pI_distribution()

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  bsa_path <- system.file("extdata", "P02769.fasta", package = "pepVet")
  mc0 <- evaluate_digest(bsa_path, enzyme = "trypsin", missed_cleavages = 0)
  mc1 <- evaluate_digest(bsa_path, enzyme = "trypsin", missed_cleavages = 1)
  p <- plot_missed_cleavage_impact(list("MC=0" = mc0, "MC=1" = mc1))
  print(p)
}
```

---

plot_mz_distribution *Precursor m/z Distribution*

Description

plot_mz_distribution() draws overlapping density fills for precursor m/z values at charge states $z = +2$ and $z = +3$, overlaid on a shaded instrument scan window. It adds a scan-window view alongside the package's length, GRAVY, and pI summaries.

Usage

```
plot_mz_distribution(
  result,
  scan_range = c(350, 1500),
  charge_states = 2:3,
  length_range = c(7L, 25L),
  show_rug = TRUE,
  title = NULL
)
```

Arguments

result	Accepted inputs: • A named list returned by evaluate_digest(). Valid peptides are extracted automatically using length_range, and m/z values are computed via calculate_peptide_mass(). • A named list of evaluate_digest() results (multi-input mode). Produces a faceted plot with one panel per result. • A data.frame / tibble with a peptide column. m/z values are computed from sequences.

- A data.frame / tibble with `mz` and `charge_state` columns (pre- computed m/z values; `charge_states` argument is ignored). If `NULL` or an unrecognised type, raises an error.

`scan_range` Numeric vector of length 2 giving the instrument's MS1 scan window boundaries in m/z. Defaults to `c(350, 1500)` (typical DDA on Orbitrap / Q-TOF instruments). Use `c(400, 1000)` for targeted methods. If `NULL` or malformed, raises an error.

`charge_states` Integer vector of charge states to compute. Defaults to `2:3`. Ignored when `result` already contains an `mz` column.

`length_range` Integer vector of length 2. Valid peptide length window used to filter input peptides. Defaults to `c(7L, 25L)`. Read from `result$params` when a full `evaluate_digest()` result is supplied.

`show_rug` Logical. When `TRUE` (default) a rug of individual peptide m/z values is added below the density fills at `alpha = 0.30`. If `NULL`, treated as `FALSE`.

`title` Optional character string for the plot title. Auto-generated when `NULL` (default).

Details

Peptides inside the active length window can still fall outside a selected MS1 scan window at the displayed charge states. The plot shows those values against the user-supplied window.

The function accepts four input types with the following precedence: (1) named list of `evaluate_digest()` results (multi-input mode, produces a faceted panel per result), (2) single `evaluate_digest()` result, (3) data.frame with a `peptide` column (m/z computed from sequences), (4) data.frame with pre-computed `mz` and `charge_state` columns.

Value

A `ggplot` object showing overlapping density fills of precursor m/z values at each charge state, with shaded instrument scan window and per-charge-state window-coverage annotations.

See Also

`evaluate_digest()` for the upstream digestion step, `calculate_peptide_mass()` for the underlying m/z calculation.

Other plot-distribution: `plot_gravy_landscape()`, `plot_length_distribution()`, `plot_missed_cleavage_impact`
`plot_pI_distribution()`

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  bsa_path <- system.file("extdata", "P02769.fasta", package = "pepVet")
  res <- evaluate_digest(bsa_path, enzyme = "trypsin")
  p <- plot_mz_distribution(res)
  print(p)
}
```

plot_peptide_overlap_map *Amino-Acid Peptide Overlap Map*

Description

`plot_peptide_overlap_map()` renders a wrapped residue-level view of a single protein and colors each amino-acid tile by how many peptides cover that residue (MC=0 only by default). Uncovered residues are white; increasing overlap depth gets progressively stronger blue fills. The subtitle reports the maximum overlap, which helps gauge whether the 6-color scale (0, 1, 2-3, 4-7, 8-15, 16+) is sufficient for your data.

Usage

```
plot_peptide_overlap_map(
  result,
  length_range = c(7L, 25L),
  missed_cleavages = 1L,
  residues_per_line = 50L,
  title = NULL
)
```

Arguments

`result`	A named list returned by `evaluate_digest()`. Must describe a single protein. If `NULL` or invalid, raises an error.
`length_range`	Optional integer vector of length 2 defining which peptide lengths count toward overlap. Defaults to `c(7L, 25L)`. Peptides are counted from the MC level specified by `missed_cleavages`. Use `length_range = NULL` to count all digested peptides at all missed-cleavage levels.
`missed_cleavages`	Integer. Which missed-cleavage level to use for the overlap count. Defaults to `1L` (standard bottom-up practice). Pass `0L` for strict non-overlapping fragments, or `NULL` to count all MC levels (automatically set when `length_range = NULL`).
`residues_per_line`	Positive integer. Number of residues to show before wrapping to the next display row. Defaults to `50L`. If `NULL` or non-integer, raises an error.
`title`	Optional character string for the plot title. Auto-generated from the protein accession and enzyme when `NULL` (default).

Details

The plotted letters are reconstructed from the MC=0 peptide set in `result$peptides`, so the plot can be generated directly from an `evaluate_digest()` result without re-reading the original FASTA input. Overlap counts are computed from valid-length MC=1 peptides by default (configurable via `missed_cleavages`). Pass `missed_cleavages = 0L` for strict non-overlapping fragments, or `length_range = NULL` to count all digested peptides at all missed-cleavage levels.

Value

A ggplot object showing one tile per residue colored by peptide overlap depth (white for uncovered, escalating blue fills for 1, 2-3, 4-7, 8-15, or 16+ covering peptides).

See Also

evaluate_digest() for the upstream digestion step.

Other plot-single: plot_cleavage_map(), plot_coverage_map(), plot_digest_profile(), plot_weight_sensitiv

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  bsa_path <- system.file("extdata", "P02769.fasta", package = "pepVet")
  res <- evaluate_digest(bsa_path, enzyme = "trypsin", missed_cleavages = 1)
  p <- plot_peptide_overlap_map(res)
  print(p)
}
```

---

plot_pI_distribution *pI Distribution: Histogram of Peptide Isoelectric Points*

Description

plot_pI_distribution() draws a histogram of peptide isoelectric points coloured by SCX fraction bin (e.g., pH 3-4, 4-5, ...) to preview fractionation outcomes. Vertical boundary lines and per-fraction count annotations are optionally overlaid.

Usage

```
plot_pI_distribution(
  result,
  fraction_breaks = c(3, 4, 5, 6, 7, 8, 9, 10),
  show_fraction_lines = TRUE,
  title = NULL
)
```

Arguments

result	Accepted inputs: • A named list returned by evaluate_digest(). pI values are computed automatically from the valid-peptide sequences. • A tibble returned by score_peptides() with include_pI = TRUE (contains a pI list column). • A plain data.frame / tibble with a numeric pI column. • A bare numeric vector of pI values. If NULL, raises an error.
fraction_breaks	Numeric vector of pH boundary values defining the fraction bins. Defaults to c(3, 4, 5, 6, 7, 8, 9, 10), which produces eight bins: <3, 3-4, ..., 9-10, >10. If NULL, raises an error.

show_fraction_lines
: Logical. When TRUE (default) vertical dashed lines are drawn at each interior fraction boundary. If NULL, treated as FALSE.

title
: Optional character string for the plot title. Auto-generated when NULL (default).

Details

The function accepts four input types with the following precedence: (1) named list of evaluate_digest() results (multi-input mode, produces overlaid density curves per result), (2) single evaluate_digest() result, (3) data.frame with a pI column, (4) raw numeric vector of pI values. When a full evaluate_digest() result is supplied, pI values are computed from valid-peptide sequences via calculate_pI().

Value

A ggplot object showing a histogram of isoelectric points coloured by SCX fraction bin with optional fraction boundary lines.

See Also

evaluate_digest() and score_peptides() for upstream peptide scoring.

Other plot-distribution: plot_gravy_landscape(), plot_length_distribution(), plot_missed_cleavage_impact plot_mz_distribution()

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  bsa_path <- system.file("extdata", "P02769.fasta", package = "pepVet")
  res <- evaluate_digest(bsa_path, enzyme = "trypsin")
  p <- plot_pI_distribution(res)
  print(p)
}
```

---

plot_proteome_overview

Proteome digest overview

Description

plot_proteome_overview() produces a three-panel portrait of a single-enzyme proteome digest from batch_evaluate():

Usage

```
plot_proteome_overview(batch, title = NULL)
```

Arguments

batch
: A tibble returned by batch_evaluate(), with columns protein_id, composite_score, verdict, the five component score columns, and the four difficulty flag columns. A proteome-aware batch may also contain S_unique, which is included in the component profile. If NULL or empty, raises an error.

title
: Optional character title for the combined figure. When NULL (default), generates an auto-title with protein count.

Details

- **(A) Score distribution:** histogram of composite scores, verdict-colored, with background zone shading for Good (>= 0.65), Moderate (0.40-0.65), and Poor (< 0.40) regions. A percentage badge is anchored in the Good zone.
- **(B) Component profile:** horizontal lollipop chart showing the median of each component score across all proteins. Each component uses its designated color from the pepVet component palette. Immediately reveals which digest quality dimension is limiting the proteome.
- **(C) Difficulty flags:** 100% stacked horizontal bars showing the proportion of proteins carrying each difficulty flag (short protein, no valid peptides, hydrophobic, low-complexity). Flags are ordered by prevalence (most common at top).

When the input batch lacks difficulty flag columns, panel C shows an empty placeholder message instead of stacked bars.

Value

A `patchwork` object with three panels: score distribution histogram (A), component median lollipop chart (B), and difficulty-flag prevalence bars (C).

Limitations

At proteome sizes exceeding 2000 proteins the figure panels become dense and text labels may overlap. Consider filtering the batch to a representative subset or increasing the output dimensions.

See Also

`batch_evaluate()` for the upstream evaluation step.

Other plot-batch: `plot_batch_comparison()`

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE) &&
  requireNamespace("patchwork", quietly = TRUE)) {
  small <- system.file(
    "extdata", "small_proteome_50_proteins.fasta",
    package = "pepVet"
  )
  batch <- batch_evaluate(small, enzyme = "trypsin")
  plot_proteome_overview(batch)
}
```

`plot_score_diagnostics` *Plot score diagnostics*

Description

`plot_score_diagnostics()` visualises the result of `score_diagnostics()` as a three-panel figure: (A) VIF bar chart with threshold lines at 5 and 10, (B) PCA scree plot with cumulative variance line and an 80% reference, (C) ablation waterfall showing the mean composite drop with error bars when each component is set to zero, annotated with the number of verdicts that flip.

Usage

```
plot_score_diagnostics(x, title = NULL)
```

Arguments

x	A list returned by score_diagnostics(). If NULL or missing required elements, raises an error.
title	Optional character string for an overall figure title. When NULL (default), generates "Score Diagnostics" with the protein and component counts.

Value

A patchwork object with three panels: VIF (A), PCA (B), ablation (C).

Limitations

Shows what score_diagnostics() returns. Interpretation needs domain knowledge.

See Also

score_diagnostics() for the upstream analysis.

Other diagnostics: score_diagnostics()

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE) &&
    requireNamespace("patchwork", quietly = TRUE)) {
  small_fasta <- system.file(
    "extdata", "small_proteome_50_proteins.fasta",
    package = "pepVet"
  )
  batch <- batch_evaluate(small_fasta, enzyme = "trypsin")
  diag <- score_diagnostics(batch)
  p <- plot_score_diagnostics(diag)
  print(p)
}
```

---

plot_weight_sensitivity

Plot weight sensitivity

Description

plot_weight_sensitivity() visualises the result of sensitivity_analysis(). For single-protein input it draws one composite density over explicit verdict zones, with threshold lines, a simulation interval, and a rug mark at the reference composite. For batch input it draws a faceted histogram of per-row verdict instability with per-enzyme mean and median annotations.

Usage

```
plot_weight_sensitivity(x, title = NULL)
```

Arguments

x	A single-protein or batch result returned by sensitivity_analysis(). Multi-enzyme summary results do not contain the per-draw values required by this plot. If NULL or unrecognised, raises an error.
title	Optional character string for the plot title. When NULL (default) a title is auto-generated.

Value

A ggplot object: density plot of composite scores with verdict shading (single-protein mode) or faceted histogram of verdict instability (batch mode).

See Also

sensitivity_analysis() for the upstream weight-sensitivity simulation.

Other plot-single: plot_cleavage_map(), plot_coverage_map(), plot_digest_profile(), plot_peptide_overlap

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE) &&
    requireNamespace("withr", quietly = TRUE)) {
  bsa_path <- system.file("extdata", "P02769.fasta", package = "pepVet")
  res <- evaluate_digest(bsa_path, enzyme = "trypsin")
  sens <- withr::with_seed(
    42,
    sensitivity_analysis(res, n_iter = 1000L)
  )
  plot_weight_sensitivity(sens)
}
```

---

recommend_enzyme *Return the highest-scoring enzyme for a single protein*

Description

recommend_enzyme() calls compare_digests() and returns the name of the enzyme with the highest composite score. When two or more enzymes are tied, all tied enzyme names are returned in alphabetical order. It is a compact model-ranking result for scripted triage that stays aligned with compare_digests().

Usage

```
recommend_enzyme(
  sequence,
  enzymes = c("trypsin", "lysc"),
  missed_cleavages = 1L,
  proteome = NULL,
  weights = NULL,
  ...
)
```

Arguments

sequence	A single-protein input passed to compare_digests(). If NULL or empty, raises an error.
enzymes	Character vector of unique enzyme names to compare. Defaults to c("trypsin", "lysc").
missed_cleavages	Maximum missed cleavages. Defaults to 1L.
proteome	Optional proteome digest tibble for uniqueness scoring. When NULL (default), no uniqueness scoring.
weights	Optional scoring weight vector. When NULL (default), uses pepVet's default scoring weights.
...	Additional scoring arguments passed to compare_digests() and ultimately to evaluate_digest() and score_peptides().

Value

A character vector of one or more enzyme names. Length greater than one only when top scores are tied within floating-point tolerance.

Limitations

Single-protein only. When multiple enzymes tie within tolerance, all are returned in alphabetical order with no further tie-breaking. The result is not an experimental recommendation.

See Also

compare_digests(), evaluate_digest()

Other evaluation: batch_compare_enzymes(), batch_evaluate(), compare_digests(), evaluate_digest(), sensitivity_analysis(), summarize_batch(), triage_proteins()

Examples

```
bsa_path <- system.file("extdata", "P02769.fasta", package = "pepVet")
recommend_enzyme(bsa_path, enzymes = c("trypsin", "lysc"))
```

---

score_diagnostics *Score diagnostics for pepVet scoring models*

Description

score_diagnostics() runs three analyses on a batch_evaluate() result to quantify multicollinearity, dimensionality, and component contributions in the scoring model. Use it to validate that the five (or six) component scores are measuring orthogonal dimensions and that each contributes meaningfully to the composite.

Usage

```
score_diagnostics(batch_result, weights = NULL)
```

Arguments

batch_result	A tibble returned by batch_evaluate(). Must contain at least two rows and two component-score columns (prefixed S_). If NULL, too few rows, or missing required columns, raises an error.
weights	Optional named numeric vector of component weights. When NULL (default), weights are inferred from the detected component columns using pepVet's default scoring weights (protein-only or proteome-aware, depending on presence of S_unique).

Details

Three analyses are performed:

VIF (Variance Inflation Factor): For each component, a linear regression is fit using all other components as predictors. VIF = 1 / (1 - R^2). VIF < 5 is acceptable. VIF > 10 indicates problematic collinearity that may warrant component removal or merger. VIF requires more than n_components + 1 observations; when this condition is not met for an input with at least two rows, NA values are returned with a warning. Perfect collinearity is represented by Inf rather than an unclassed regression warning.

PCA (Principal Component Analysis): Centered and scaled PCA on the component-score matrix. The proportion of variance explained by each principal component reveals the effective dimensionality of the scoring model. If PC1 + PC2 explain > 80% of variance, the model has low effective dimensionality (expected for a well-designed multi-criteria score).

Ablation: Each component is set to 0 (its minimum possible value) for all proteins while others remain at their actual values, and the composite score is recomputed. The mean and maximum drop in composite score, plus the number of verdicts that flip, quantify each component's marginal contribution. Components with mean drops near zero are candidates for removal or down-weighting.

Value

A named list with six elements:

vif A named numeric vector of VIF values, one per component. NA when too few proteins for reliable estimation and Inf for perfect collinearity.

pca A list with var_explained (numeric vector), loadings (rotation matrix), sdev (standard deviations), and x (PCA scores matrix).

ablation A data.frame with columns component, weight, mean_drop, sd_drop, max_drop, and n_verdict_flipped.

n_proteins Number of proteins in the input.

n_components Number of component scores detected.

weights The resolved weight vector used for ablation.

Limitations

Diagnostics are meaningful only on batch results with enough proteins (at least 10-20 recommended; VIF requires more than n_components + 1). Results are specific to the enzyme, missed-cleavage setting, and protein set used. Running diagnostics on a different batch may give different VIF and ablation values.

See Also

batch_evaluate() for upstream batch evaluation, plot_score_diagnostics() for visualization.

Other diagnostics: plot_score_diagnostics()

Examples

```
small_fasta <- system.file(
  "extdata", "small_proteome_50_proteins.fasta",
  package = "pepVet"
)
batch <- batch_evaluate(small_fasta, enzyme = "trypsin")
diag <- score_diagnostics(batch)
diag$vif
diag$ablation

# Proteome-aware mode
prot_digest <- digest_protein(small_fasta, enzyme = "trypsin")
batch_pa <- batch_evaluate(small_fasta, enzyme = "trypsin",
  proteome = prot_digest)
diag_pa <- score_diagnostics(batch_pa)
diag_pa$vif
```

---

score_peptides *Score digested peptides for pre-acquisition review*

Description

score_peptides() summarizes a pepVet digest tibble into per-protein scoring components and a weighted composite score. The scoring model is designed for digest planning and enzyme comparison, not for post-search peptide detectability prediction.

Usage

```
score_peptides(
  digest_result,
  proteome = NULL,
  weights = NULL,
  gravy_range = c(-1, 0.6),
  length_range = c(7L, 25L),
  enzyme = "trypsin",
  include_pI = FALSE
)
```

Arguments

digest_result	A digest tibble produced by digest_protein() or an equivalent table containing the columns protein_id, peptide, start, end, length, and missed_cleavages. If NULL or missing required columns, raises an error.
proteome	Optional digest tibble representing the comparison proteome used for peptide uniqueness scoring. When NULL (default), S_unique is excluded and protein-only default weights are used.

weights	Optional numeric weight vector. In protein-only mode the default weights are c(S_length = 0.200, S_coverage = 0.348, S_count = 0.226, S_hydro = 0.138, S_charge = 0.088). In proteome-aware mode the weights are c(S_length = 0.160, S_coverage = 0.279, S_count = 0.181, S_hydro = 0.110, S_charge = 0.070, S_unique = 0.200). The weight vectors are documented expert priors chosen for pepVet, not coefficients fitted to experimental outcomes. Coverage and peptide count receive the largest combined weight in the default protein-only vector.
gravy_range	Numeric vector of length 2 defining the inclusive GRAVY range used by S_hydro. Defaults to c(-1.0, 0.6). If NULL, raises an error.
length_range	Integer vector of length 2 defining the inclusive valid peptide length range used by S_length, S_coverage, S_count, S_hydro, S_charge, and S_unique. Defaults to c(7L, 25L). If NULL, raises an error.
enzyme	Cleavage rule name used to choose the fallback expected peptide length when the digest contains fewer than three peptides. Defaults to "trypsin". If NULL, raises an error.
include_pI	Logical flag indicating whether to append a pI list column containing peptide-level pI values for valid peptides. Defaults to FALSE. If NULL, raises an error.

Details

Valid peptides are defined as peptides with lengths between 7 and 25 residues inclusive by default, but this window can be changed with length_range. Coverage is computed from valid peptide coordinates with overlapping intervals reduced before coverage is summed. S_hydro uses the inclusive gravy_range; mixed sequences average known hydrophobicity values, and sequences with no known values score zero. S_unique is only computed when a proteome digest is supplied. S_charge measures the fraction of valid peptides that contain at least one non-terminal basic residue to capture extra charge-state richness rather than baseline ionizability. A zero S_count means that the enzyme produced no cleavage sites or no usable peptides. In that case pepVet applies a hard failure: composite_score is set to zero and the verdict is Poor, regardless of the other component values. This is the documented exception to the weighted-sum equation. Composite verdicts are classified as Good for scores >= 0.65, Moderate for scores >= 0.4, and Poor otherwise.

Value

A tibble with one row per protein_id and the component score columns S_length, S_coverage, S_count, S_hydro, S_charge, optional S_unique, plus composite_score, verdict, and median_peptide_length, and preset_used. The median_peptide_length column records the digest-level denominator used in the enzyme-aware S_count calculation. The preset_used column records the named preset whose resolved scoring configuration exactly matches the current call, or "custom" otherwise. When include_pI = TRUE, the output also includes a pI list column with one named numeric vector per protein, storing valid-peptide pI values keyed by peptide sequence.

Limitations

pepVet is a rule-based digest-ranking model, not a peptide detectability predictor. The verdict thresholds are heuristic, the default and preset weights are expert priors rather than empirically fit coefficients, and the scoring windows assume conventional C18 reversed-phase LC with ESI. pepVet does not model PTMs, chemical labels, chromatographic gradients, or instrument-specific fragmentation behavior. Composite scores are interpretable rankings with no physical unit; they are not calibrated probabilities. Cross-workflow comparisons are only meaningful when the resolved scoring configuration matches, which is why preset_used is recorded in the output.

Examples

```
digest_result <- digest_protein("MKWVTFISLLFLFSSAYSR")
score_peptides(digest_result)

target_and_background <- digest_protein(
  c(target = "AAAAAAARAAAAAAAK", background = "AAAAAAARGGGGGGGK")
)
score_peptides(
  target_and_background[target_and_background$protein_id == "target", ],
  proteome = target_and_background
)
```

`sensitivity_analysis` *Weight sensitivity analysis*

Description

`sensitivity_analysis()` perturbs the resolved scoring weights using a Dirichlet distribution and reports how often the verdict or enzyme ranking changes. It compares each perturbation with the stored reference score and applies the scoring hard-fail rule consistently.

Usage

```
sensitivity_analysis(
  x,
  nu = 63,
  n_iter = 2000L,
  chunk_size = 100L,
  importance = FALSE,
  corner_cases = FALSE
)
```

Arguments

`x`	An `evaluate_digest()`, `compare_digests()`, `batch_evaluate()`, or `batch_compare_enzymes()` result.
`nu`	Dirichlet concentration scaling factor. Controls how much the perturbed weights are allowed to vary from the resolved reference weights. Must be at least `0.1`. Default `63` gives a standard deviation of approximately 0.05 for a weight of 0.2. Larger values produce tighter distributions. Smaller values allow more variation.
`n_iter`	Number of Monte Carlo iterations. Default `2000L`.
`chunk_size`	Number of score rows to process per chunk in batch mode. Default `100L` bounds the largest score-by-iteration matrix.
`importance`	Logical. If `TRUE`, compute the squared Pearson correlation between each sampled weight and the composite score across iterations. These marginal associations are descriptive, can reflect dependence among Dirichlet weights, and do not form a variance decomposition.
`corner_cases`	Logical. If `TRUE`, report the composite score when each weight is at its exact 95 percent marginal Dirichlet interval bound. Other weights retain their reference proportions. Corner diagnostics are available for single-protein and multi-enzyme inputs, not batch inputs.

Details

The function draws from the current R random-number generator and advances its state. It does not set or restore a seed. Callers control reproducibility outside the function.

Value

A mode-dependent list:

- Single-protein output contains `iterations`, `convergence`, `summary`, and `settings`. `iterations` contains sampled weights, `iteration`, `composite_score`, and `verdict`. `convergence` contains `iteration`, `cumulative_stability`, and `mc_se`. `summary` contains `verdict_pct`, `composite_ci`, `composite_mean`, `reference_composite`, `reference_weights`, `reference_verdict`, and requested diagnostics.
- Multi-enzyme output contains `summary` and `settings`. `summary` contains `top1_stability`, `kendall_mean`, `kendall_defined_fraction`, `reference_composites`, and requested per-enzyme diagnostics.
- Batch output contains `per_protein`, `summary`, and `settings`. `per_protein` contains `protein_id`, `reference_composite`, `verdict_instability`, `composite_mean`, `composite_lo`, `composite_hi`, and remaining input columns. `summary` contains `total_instability`, `mean_ci_width`, `ci_width_quantiles`, `reference_weights`, and requested `weight_importance`.

Every `settings` list records `distribution`, `nu`, `n_iter`, applicable `chunk_size`, `reference_weights`, `fixed_zero_weights`, and `verdict_thresholds`.

Interpreting results

Monte Carlo estimates are approximate, and their precision depends on `n_iter`. Tied top scores share credit equally. Kendall correlation is undefined when the reference or sampled scores have only one rank.

Limitations

The analysis perturbs only weights within the Dirichlet framework. It does not test alternative scoring functions, verdict thresholds, or parameter ranges. The distribution is a package-chosen computational stress test, not an empirical uncertainty model. A reference weight of zero remains zero in every draw. Stability does not establish that a weight or verdict is biologically correct.

See Also

Other evaluation: `batch_compare_enzymes()`, `batch_evaluate()`, `compare_digests()`, `evaluate_digest()`, `recommend_enzyme()`, `summarize_batch()`, `triage_proteins()`

Examples

```
if (requireNamespace("withr", quietly = TRUE)) {
  bsa_path <- system.file("extdata", "P02769.fasta", package = "pepVet")
  res <- evaluate_digest(bsa_path, enzyme = "trypsin")
  sens <- withr::with_seed(
    42,
    sensitivity_analysis(res, n_iter = 1000L)
  )
  sens$summary$verdict_pct
}
```

summarize_batch *Summarize a batch digest evaluation*

Description

summarize_batch() extracts aggregate statistics from a batch_evaluate() result tibble. Returns a named list with verdict distribution, score distribution, per-component averages, the lowest-scoring proteins, and a heuristic set of enzyme-switch candidates.

Usage

```
summarize_batch(batch_result)
```

Arguments

batch_result	A tibble returned by batch_evaluate(). If NULL or empty, raises an error.

Details

enzyme_switch_candidates is a heuristic flag list derived from sequence-level difficulty flags, not from running alternative enzymes. Use compare_digests() to confirm whether a specific alternative enzyme improves the verdict for a flagged protein.

Value

A named list with five elements:

verdict_counts A tibble with columns verdict, n, and pct covering the three verdict categories.

score_distribution A named numeric vector with mean, median, sd, q25, q75, min, and max of composite scores.

component_summary A named numeric vector of per-component mean scores. The lowest values identify the weakest scoring dimension across the proteome.

problem_proteins A tibble of proteins in the bottom 10% by composite score, ordered ascending, with all score and flag columns.

enzyme_switch_candidates A tibble of Moderate or Poor proteins where flag_hydrophobic or flag_short_protein is TRUE. These rows are candidates for direct comparison with another enzyme or preset.

Limitations

The enzyme_switch_candidates are heuristic flags based on sequence difficulty, not actual re-evaluation with alternative enzymes. Use compare_digests() to confirm whether a switch improves the verdict.

See Also

batch_evaluate(), triage_proteins()

Other evaluation: batch_compare_enzymes(), batch_evaluate(), compare_digests(), evaluate_digest(), recommend_enzyme(), sensitivity_analysis(), triage_proteins()

Examples

```
small_path <- system.file(
  "extdata", "small_proteome_50_proteins.fasta",
  package = "pepVet"
)
batch <- batch_evaluate(small_path, enzyme = "trypsin")
summary <- summarize_batch(batch)
summary$verdict_counts
summary$component_summary
```

`triage_proteins` *Triage proteins from a batch evaluation*

Description

`triage_proteins()` appends an `action` column to the flat tibble returned by `batch_evaluate()` with deterministic action labels based on each protein's verdict and difficulty flags.

Usage

```
triage_proteins(batch_result)
```

Arguments

`batch_result`	A tibble returned by `batch_evaluate()`. If `NULL` or empty, raises an error.

Value

A tibble with one row per protein containing all score and flag columns from the flat batch summary, plus an `action` column. Possible values:

`"proceed"` Good verdict. No intervention indicated.

`"consider_alternative"` Moderate verdict without a sequence-level difficulty flag. Review component scores and consider a preset or missed-cleavage adjustment.

`"try_other_enzyme"` Moderate or Poor verdict with `flag_hydrophobic` or `flag_short_protein`, or any Poor verdict without an intrinsic complexity flag. This action marks the protein for an explicit alternative-enzyme comparison.

`"skip"` No valid peptides or a low-complexity sequence. This action marks the row for manual review rather than asserting that another enzyme cannot help.

Limitations

Triage actions are advisory, based on heuristic difficulty flags from the batch score columns. They do not re-evaluate the protein with alternative enzymes. Use `compare_digests()` for that.

See Also

`batch_evaluate()`, `summarize_batch()`

Other evaluation: `batch_compare_enzymes()`, `batch_evaluate()`, `compare_digests()`, `evaluate_digest()`, `recommend_enzyme()`, `sensitivity_analysis()`, `summarize_batch()`

Examples

```
small_path <- system.file(
  "extdata", "small_proteome_50_proteins.fasta",
  package = "pepVet"
)
batch <- batch_evaluate(small_path, enzyme = "trypsin")
triaged <- triage_proteins(batch)
table(triaged$action)
```

Index