Supporting Information - 1 - Material Methods: *Complete Simulation of timsTOF PASEF Raw Datasets with Timsim Enables Precise Benchmarking of False Discovery and Phosphosite Localization Error Rates*

David Teschner[1,2], Zixuan Xiao[3], Tim Maier[1,2], David Gomez-Zepeda[4,5], Mateusz K. Łącki[6], Michał P. Startek[7], Ute Distler[6], Tanja Ziesmann[6], Mathias Wilhelm[3,8], Andreas Hildebrandt[1,2], and Stefan Tenzer[4,5,6]

[1]Institute of Computer Science, Johannes Gutenberg University, 55128 Mainz, Germany
[2]Institute for Quantitative and Computational Biosciences (IQCB), Johannes Gutenberg University, 55128 Mainz, Germany
[3]Computational Mass Spectrometry, School of Life Sciences, Technical University of Munich, Freising 85354, Germany
[4]Helmholtz Institute for Translational Oncology, Mainz, Germany
[5]German Cancer Research Center (DKFZ), 69120 Heidelberg, Germany
[6]University Medical Center, Johannes Gutenberg University, 55131 Mainz, Germany
[7]Department of Mathematics, Informatics, and Mechanics, University of Warsaw, 02-097 Warsaw, Poland
[8]Munich Data Science Institute (MDSI), Technical University of Munich, Garching 85748, Germany

March 3, 2026

1 Material and Methods

1.1 Hardware

Generation of *in silico* datasets was performed on a workstation computer running Linux (Ubuntu 22.04), equipped with a AMD Ryzen CPU with 8 logical cores (16 threads), 32 GB of RAM, and an NVIDIA RTX 2070 GPU, as well as on a workstation running Linux (Ubuntu 22.04), equipped with an AMD Ryzen Threadripper CPU and 64 logical cores (128 threads), 256 GB of RAM, and an NVIDIA RTX 4090 GPU.

1.2 Software Development

All software was developed using Rust (version 1.88.0) and Python (version 3.11). The Rust library PyO3 (version 0.22.1) was utilized to expose Rust functionalities to Python. The software depends on a variety of other libraries; a detailed list of these dependencies can be found in the `rustims` project repository on GitHub.

1.3 Raw Datasets

A collection of real reference datasets were recorded both in DDA and DIA mode to serve as a source of realistic background noise. To avoid inflating false positives, all blank samples were searched by all benchmarked software tools using the same parameters per software reported for the hela complexity ramp experiment. IDs found in a background sample were always subtracted from the reported findings of simulated datasets.

1.3.1 DIA Blank Samples

To record blank runs in dia-PASEF mode (PMID: 33257825), 1 µL of 0.1% (v/v) formic acid (FA) in LC-MS grade water was injected in triplicates on a nanoElute LC system (Bruker Corporation, Billerica, MA, USA) equipped with a reversed phase C18 column (PepSepTM, 25 cm x 150 µm 1.5 µm) which was heated to 40°C. Blank samples were loaded onto the analytical column in direct injection mode at 800 bar. Mobile phase A was 0.1% FA (v/v) in water and mobile phase B 0.1% FA (v/v) in ACN. A non-linear gradient from 2% to 38% mobile phase B over 26.5 min at a flow rate of 1 µL/min was applied. Afterwards, column was rinsed for 4.5 min at 95% B. The blank samples were analyzed in positive mode ESI-MS on a timsTOF Pro2 mass spectrometer (Bruker Corporation) in dia-PASEF mode (PMID: 33257825). A Captive Spray source was used for ionization, with a capillary voltage of 1600 V, dry gas at 3.0 L/min, and dry temperature at 200°C and a TIMS-in pressure of 2.7 mbar. The dual TIMS was operated at a fixed duty cycle close to 100% using equal accumulation and ramp times of 100 ms each spanning a mobility range from 1/K0 = 0.6 Vs cm-2 to 1.6 Vs cm-2. We defined 36×25 Th isolation windows from m/z 300 to 1,165 resulting in 2-3 dia-PASEF scans per TIMS cycle and an overall cycle time of 1.7 s. The collision energy was ramped linearly as a function of the mobility from 59 eV at 1/K0 = 1.3 Vs cm-2 to 20 eV at 1/K0 = 0.85 Vs cm-2. MS1 and fragment ion spectra were recorded with a mass range spanning from m/z 100-1,700.

For the acquisition of blank runs in midia-PASEF mode [**?**], 1 µL of 0.1% (v/v) formic acid (FA) in LC-MS grade water was analyzed on a nanoElute LC system (Bruker Corporation, Billerica, MA, USA) coupled to a timsTOF HT mass spectrometer (Bruker Corporation). Blank samples were loaded at 600 bar onto a reversed phase C18 column (Aurora UHPLC emitter column, 25 cm x 75 µm 1.7 µm, 120 Å pore size, IonOpticks, Australia) in in direct injection mode. Column was heated to 50°C. Mobile phase A was 0.1% FA (v/v) in water and mobile phase B 0.1% FA (v/v) in ACN. Blank samples were separated running a non-linear gradient from 2% to 37% mobile phase B over 38 min at a flow rate of 400 nL/min. Afterwards, column was rinsed for 5 min at 95% B. A Captive Spray source was used for ionization, with a capillary voltage of 1600 V, dry gas at 3.0 L/min, dry temperature at 180 °C, and TIMS-in pressure of 2.7 mbar. MS data were acquired in midia-PASEF mode as previously described [**?**]. For midia-PASEF acquisition, we programmed sixteen scans, i.e. MIDIA frames, with quadrupole isolation window widths of 36 Th (after quadrupole calibration for scanning mode) resulting in an overall cycle time of around 1.8 s. Quadrupole isolation window positions were shifted 12 Th (as compared to the previous frame) during acquisition resulting in an overlap of 24 Th between two neighboring frames. Precursor mass range covered m/z = 120 – 1256 Th. Another blank sample was acquired using quadrupole isolation window widths of 24 Th and a window overlap of 12 Th covering a precursor mass range of m/z = 120 - 1244 Th. The dual TIMS was operated at a fixed duty cycle close to 100% using equal accumulation and ramp times of 100 ms each spanning a mobility range from 1/K0 = 0.7 Vs cm-2 to 1.3 Vs cm-2. The collision energy was ramped linearly as a function of the mobility from 70 eV at 1/K0 = 1.6 Vs cm-2 to 20 eV at 1/K0 = 0.60 Vs cm-2. A fixed value of 10 eV was set for experiments, where no peptide fragmentation was induced and unfragmented precursors were recorded.

1.3.2 DDA Blank Samples

Two DDA Blank samples were acquired by LC-MS: For HLAI data, a blank was analyzed using a nanoElute coupled to a timsTOF-Pro-2 mass spectrometer. Data was acquired using Compass Hystar (Bruker) version 5.1, and timsControl version 4.0.5 (Bruker). For each replicate, 1 µL of 0.1% formic acid in water (HPLC-MS grade) was directly injected in a C18 Reversed-phase (RP) Aurora 25 cm analytical column (25 cm x 75 µm ID, 120 Åpore size, 1.7 µm particle size, IonOpticks, Australia) and separated using a 47 min gradient increasing the proportion of phase B (ACN with 0.1% FA (v/v)) to phase A (water with 0.1% FA (v/v)) from 2% to 37% B. A Captive Spray source was used for ionization, with a capillary voltage of 1600 V, dry gas at 3.0 L/min, dry temperature at 180 °C, and TIMS-in pressure of 2.7 mBar. MS data were acquired in DDA-PASEF mode as previously described [**?**] using a 100 ms TIMS ramp, 10 MS2 frames/cycle, 3 cycle overlap and high-sensitivity mode. To generate datasets with MS2 spectra of the background over the whole 1/K_0 - m/z plan, the isolation polygon was removed and ions with charge 1 to 4 were included for fragmentation.

For HeLA DDA data, a blank was analyzed using an Evosep One coupled with a timsTOF Ultra mass spectrometer. Data was acquired using Compass Hystar (Bruker) version 6.3.1.8, and timsControl version 5.1.8.0 (Bruker). For each replicate, 20 µL of 0.1% formic acid in water (HPLC-MS grade) were loaded into an Evotip Pure, following the recommendations of the provider (Evosep). Then, a 30 samples per day (SPD) standard method was used to analyze the sample. A, EvoSep Performance C18 column (15 cm x 150 µm, 1.5 µm particle size) heated at 40°C was used for separation, connected to a 20 µm steel emitter. A CaptiveSpray Ultra source was used for ionization, with a capillary voltage of 1,500 V, dry gas at 3.0 L/min, dry temperature at 200°C, and TIMS-in pressure of 2.4 mBar. MS data were acquired in DDA-PASEF using a 100 ms TIMS ramp, 10 MS2 frames/cycle, 3 cycle overlap and high-sensitivity mode. To simulate the usual background for a proteomics experiment, the standard isolation polygon was used which results in the fragmentation of only multiply-charged ions, although the fragmentation of ions with 1 to 5 charges was enabled.

1.4 Simulated Raw Datasets

For software benchmarking across various domains, proteomics datasets that cover diverse aspects and processing challenges were created. These include a tryptic HeLa dataset, an HLA1 dataset containing HLA class I immunopeptidomics data, a HeLa dataset containing variable phosphorylation sites (HeLa-Phos), and a mixture of peptides originating from different species (HeLa, yeast, and *E. coli*, HYE). These datasets are further explained in the following sub-sections and summarized in table 1.

HeLa Samples (dia-PASEF)

For the HeLa samples captured in dia-PASEF mode, UP000005640 Uniprot human proteome fasta file version 2024/05/17 was used, which was digested tryptically *in silico*, allowing up to two missed cleavages, sampling peptides of length 7 to 30. Complexities of 5k, 10k, 25k, 50k, 125k, and 250k peptides were generated. All samples were generated with two replicates (samples that contain the same composition of peptides, ions and intensities) with a gradient length of 60 minutes, capturing an inverse ion-mobility range of 0.6 - 1.6 $1/K_0$.

Dataset ID	Acquisition	Sample Type	Complexity	Repl.	Source	Primary Use Case
DIA-H01	dia-PASEF	HeLa	5k–250k	2	In silico	ID + FDR
DIA-M01	dia-PASEF	HYE	10k–250k	2	In silico	Quantification
DIA-PH01	dia-PASEF	HeLa-Phos	5k–125k	2	In silico	PTM localization
DDA-H01	dda-PASEF	HeLa	150k-300k	12	In silico	MBR benchmark
DDA-HLA01	thunder-PASEF	HLA1	10k, 100k	3	Experimental	ID + FDR

Table 1: **Overview of simulated benchmarking datasets generated using the *timsim* framework.** The datasets were simulated in both dda-PASEF and dia-PASEF acquisition modes, covering four core tasks in raw data processing: identification, quantification, PTM site localization, and false transfer rate (FTR) estimation. Each dataset varies in sample acquisition mode, composition, and complexity (number of peptides simulated over different LC gradients). Replicates (indicated as "2", "3", etc.) correspond to the number of simulated runs per complexity. thunder-PASEF, usually written as thunder-dda-PASEF, is also a dda-method, specifically tailored to capture singly charged ions.

HeLa, Yeast, *E. Coli* (HYE) Proteome Mixture Samples (dia-PASEF)

A collection of datasets that simulate the mixing of Proteomes coming from different organisms, mimicking the protocol implemented for the evaluation of MaxDIA quantification performance [**?**] was generated. A total of 1,000 proteins were sampled at random, each from the Uniprot Proteomes UP000005640, UP000002311 and UP000000558, file version 2024/05/17, resulting in a total of 3,000 proteins. The fasta file then was used to perform tryptic digest *in silico*, allowing up to two missed cleavages, sampling peptides of length 7 to 30. Complexities of 10k, 25k, 50k, 125k, and 250k peptides were generated. During simulation, all randomly generated occurrences per peptide were additionally multiplied with an artificial dilution factor. Those dilution factors varied for the two generated datasets. For datasets-A: HeLa 0.65, Yeast 0.15, *E. Coli* 0.20. For datasets-B: HeLa 0.65, Yeast 0.30, *E. Coli* 0.05.

Phosphorylation Samples (dia-PASEF)

For the phosphorylation samples, the UP000005640 Uniprot human proteome fasta file version 2024/05/17 was used. It was digested tryptically *in silico*, allowing up to two missed cleavages, sampling peptides of length 7 to 30. Complexities of 5k, 10k, 25k, and 100k peptides were generated. Sequences that contained at least two potential sites (amino acids S, T, or Y) were selected. The replicates per complexity then varied the modification sites per raw dataset, where dataset-A was phosphorylated always at a different site compared to dataset-B.

HeLa Samples (dda-PASEF)

For the HeLa samples simulated in dda-PASEF mode, the UniProt human reference proteome (UP000005640, version 2024/05/17) was used. The proteome was digested *in silico* using trypsin, allowing up to two missed cleavages and selecting peptides between 7 and 30 amino acids in length. All samples were drawn from the same precursor ion pool consisting of 150,000 peptides. Two groups of replicates were generated: three with a 30-minute gradient and three with a 60-minute gradient. DDA acquisition was performed in topN mode, selecting one precursor frame every 7th frame. Up to 8 precursors were targeted per fragment frame, with a minimum precursor intensity of 1,000 and a dynamic exclusion window of 25 frames to avoid repeated selection. To introduce variation between replicates, Gaussian noise was added to the precursor apex values for retention time, ion mobility, and log-intensity. The standard deviations were 10 s for retention time, 0.01 for ion mobility (expressed as $1/K_0$), and 0.02 for log-intensity. All datasets captured an ion-mobility range of 0.65 - 1.76 $1/K_0$.

HLA1 Samples (thunder-dda-PASEF)

In contrast to simulations that were performed starting from a fasta file containing a reference proteome, an accurate immunopeptidomics dataset cannot be easily generated by cleaving the proteins *in silico*. Immunopeptides are generated in the cell by the antigen processing and presenting machinery, including various proteolytic events and binding to HLA complexes. This results in peptides with restricted size and sequence patterns imprinted by the individual HLA alleles. In the case of HLA1 ligands, peptides are typically 8 to 13 amino acid long, and show anchor amino acids usually at the second position and the C-terminus. To enable the simulation of an immunopeptidomics experiment, the HLA1 samples were generated starting from a collection of experimentally measured and confidently identified peptides (FDR $\leq 1\%$) as previously published [**?**, **?**]. This included peptides from three cell lines, JY (CVCL_0108), HeLa (CVCL_0030), and SK-MEL-37 (CVCL_3878); and a commercial human plasma (purchased from Zen-Bio (USA, SER-SPL). The compiled list included 113,857 stripped peptide sequences covering 17 distinct HLA binding motifs. Two datasets were generated by randomly selecting peptides with a length between 7 and 25 amino acids. The higher complexity dataset was generated with 100,000 peptides in a 60-minute gradient, while the lower complexity dataset was generated with 10,000 peptides in a 40-minute gradient. Both datasets captured an ion-mobility range of 0.65 - 1.76 $1/K_0$, using the Thunder isolation polygon for precursor fragmentation selection [**?**]. The simulation used the binomial charge state estimator, accounting for the many peptides expected to be singly charged (see supplement information *Material Methods*, section 1.1.4 *Charge State Estimation* for a detailed explanation).

1.5 External Software Tools and Post-processing

1.5.1 DIA-NN

DIA-NN v1.8.1, v1.9.2, and v2.0 were used with library free search / library generation enabled, meaning DIA-NN generated a predicted library for searching. All settings were used with default parameters, except the following changes: peptide length range was set to 7 - 30, missed cleavages was set to 2, and Protease was set to Trypsin/P. For the samples containing phosphorylation, the Phospho checkbox was enabled. An exhaustive list of settings can be found with the automatically generated log files.

1.5.2 FragPipe

FragPipe v22.0 and v23.0 were used. The *DIA_Speclib_Quant_diaPASEF* workflow was used for all dia-PASEF proteomics datasets, and the *DIA_Speclib_Quant_Phospho_diaPASEF* workflow was employed for the phosphorylation datasets. The *LFQ_MBR* workflow (v23.0) was employed for the ddaPASEF HeLa datasets for Match-Between-Run analysis with ion level FDR set to 0.01 (default parameter). The *Nonspecific-HLA* workflow is used for the HLA1 dataset. All MSFragger settings were used with default parameters, except for the following changes: Enzyme was set to Trypsin and peptide length was set to 7 - 30 amino acids. An exhaustive list of settings can be found with the automatically generated log files, see *Data Availability*.

All MSFragger settings were used with default parameters, except the following changes: Enzyme was set to Trypsin, Peptide length was set to 7 - 30 for the HeLa dataset, to 7 - 25 for the HLA1 dataset and MSBooster was turned off for the HLA1 dataset. An exhaustive list of settings can be found with the automatically generated log files, see *Data Availability*.

1.5.3 MaxQuant

MaxQuant version 2.6.7.0 was used for the analysis of dia-PASEF data, the TIMS-MaxDIA workflow was chosen with default parameters, except that enzyme was set to trypsin. MaxQuant version 2.6.2.0 was used for the dda-PASEF HeLa datasets for the Match-Between-Run analysis. An exhaustive list of settings can be found with the automatically generated log files, see *Data Availability*.

1.5.4 PEAKS

PEAKS XPro (v10.6, build 20201221) was used for the analysis of Thunder-DDA-PASEF data. The option `timstof_feature_min_charge` (in file `\algorithmpara\feature_detection_para.properties`) was set to 1 to allow the identification of singly-charged features. The protein database was composed of the UniProtKB (Swiss-Prot) reference proteomes of *Homo sapiens* (Taxon ID 9606, UP000005640, downloaded 22/April/2024), supplemented with a list of possible contaminants [**?**]. Protein *in silico* digestion was configured to unspecific cleavage and no enzyme. Methionine oxidation, cysteine cysteinylation, and Protein N-terminal acetylation were set as variable modifications. Peptides were identified with mass accuracy thresholds of 15 ppm for MS1 and 0.03 Da for MS2. Results were filtered at $\mathrm{FDR} \leq 0.01$ for peptides and $-10 \log P \geq 0$ for proteins.

1.5.5 Spectronaut

Spectronaut v20.1.250624.92449 was used. The data was analysed using library-free database search (directDIA+), with vendor provided workflows, in case of dia-PASEF data "BGS Factory Settings (default)", for the phosphorylated datasets "BGS phospho PTM workflow". In both cases the protease was changed to Trypsin and Max Peptide Length was set to 30, otherwise default parameters were used.

1.6 Post-processing

A separate Python module was generated to post-process all software outputs and combine them into a single resource table per conducted experiment. This module can be found on GitHub[1], together with a collection of Jupyter-Notebooks allowing to re-create any of the here presented results.

1.6.1 Output normalization.

Each software tool reports results in a tool-specific format with vendor-defined column names and modification notations. Before any metric is computed, all outputs are parsed into a unified schema with columns `sequence` (stripped amino acid sequence), `sequence_modified` (sequence with UNIMOD tags), `charge`, `protein_id`, `rt`, `inverse_mobility`, `intensity`, `pep`, `ion_q`, `peptide_q`, `protein_q`, `protein_group_q`, `software`, `filename`, and `experiment`. Modification notations are harmonized to the UNIMOD bracket format `[UNIMOD:N]`: DIA-NN parentheses are rebracketed and `UniMod` is uppercased; FragPipe modification masses are converted to UNIMOD identifiers; MaxQuant leading/trailing underscores are stripped and cysteines are annotated as `C[UNIMOD:4]`; Spectronaut named tags (e.g., `[Phospho (STY)]`) are replaced with their UNIMOD equivalents. Q-values are mapped to level-specific columns: DIA-NN's `Q.Value`/`Peptidoform.Q.Value`, Spectronaut's `FG.Qvalue`/`EG.Qvalue`/`PG.Qvalue`, and FragPipe's per-level probabilities are all renamed to `ion_q`, `peptide_q`, and `protein_group_q` respectively. MaxQuant reports only a single `Q-value` at the

[1]https://github.com/theGreatHerrLebert/timsim-bench

protein group level; ion- and peptide-level q-values are therefore set to the nominal threshold (0.01) for that tool, and protein-group q-values are used for all MaxQuant-specific filtering.

1.7 Benchmarking Metrics

This section provides formal definitions for all performance metrics used to evaluate software tools against the timsim ground truth. Because the simulated datasets have perfectly known ground truth, every reported identification can be unambiguously classified as a true positive (TP) or a false positive (FP), without the need for decoy-based approximations.

1.7.1 Analysis Levels and Identification Units

Identifications are evaluated at three hierarchical levels:

- **Ion level** (precursor ion): An ion is the unique pair (sequence, charge state). Evaluation is performed in two modes:
  - *Stripped sequence level*: modification annotations are removed; two ions are identical if they share the same amino acid sequence and charge state, regardless of variable modifications. This criterion is reported unless stated otherwise.
  - *Modified sequence level* (peptidoform level): the full sequence including UNIMOD modification tags is retained; two ions are identical only if they share the same modified sequence and charge state. This is the stricter criterion used to detect miscalled modifications.
- **Peptide level**: A unique amino acid sequence string (stripped of modifications).
- **Protein level**: A unique protein identifier, evaluated using the group credit rule described below.

1.7.2 True False Discovery Rate

Let $\mathcal{G}_\ell$ denote the ground truth set at level ℓ, i.e. the set of unique identifiers present in the simulated dataset. Let $\mathcal{R}_\ell$ denote the set of identifiers reported by a software tool after filtering at its nominal level-specific q-value threshold (typically $q_\ell \leq 0.01$, where q_ℓ is the ion, peptide, or protein group q-value depending on ℓ). We shall refer to the following expression 1

$$\mathrm{FDR}_\ell = \frac{|\mathcal{R}_\ell \setminus \mathcal{G}_\ell|}{|\mathcal{R}_\ell|} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TP}}, \tag{1}$$

as the true false discovery rate. Note that in statistical literature the above expression would rather be called the False Discovery Percentage, which would be a realization of a random variable and that the expectation of that variable would be False Discovery Rate proper. In context of our simulation, we exactly know the ground truth and there is no probabilistic aspect to the calculation. Otherwise said, both terms coincide and to keep the argument simple we shall continue to refer to that quantity as FDR. The same argument will be valid also for all following definitions.

where $\mathrm{TP} = |\mathcal{R}_\ell \cap \mathcal{G}_\ell|$ and $\mathrm{FP} = |\mathcal{R}_\ell \setminus \mathcal{G}_\ell|$. Throughout the paper, FDR is expressed as a percentage. The distinction from the software-reported (nominal) FDR is critical: the nominal FDR is estimated by the tool using its own scoring model and decoy strategy, whereas the true FDR is derived directly from the known simulation ground truth.

Prior to evaluation, ion-level identifications found in blank-run searches are subtracted from the reported sets of all software tools to avoid inflating FP counts with background contaminants. The same blank search parameters were used across all tools.

Protein-level group credit rule. Proteomics software typically reports protein groups rather than individual proteins. For protein-level evaluation, a group credit rule is applied: a reported protein group with candidate set C is classified as TP in its entirety if $C \cap \mathcal{G}_{\text{protein}} \neq \emptyset$ (i.e., at least one protein in the group is present in the simulated proteome). If $C \cap \mathcal{G}_{\text{protein}} = \emptyset$, all proteins in C are classified as FP. This is consistent with standard parsimony-based protein inference.

1.7.3 True Positive Rate and Sensitivity

The true positive rate (TPR), also referred to as sensitivity or recall, is defined as

$$\mathrm{TPR}_\ell = \frac{|\mathcal{R}_\ell \cap \mathcal{G}_\ell|}{|\mathcal{G}_\ell|} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}, \tag{2}$$

where $\mathrm{FN} = |\mathcal{G}_\ell \setminus \mathcal{R}_\ell|$ is the number of false negatives (simulated identifiers not recovered by the tool). TPR is also expressed as a percentage.

Intensity-stratified sensitivity is computed by partitioning $\mathcal{G}_\ell$ into bins according to simulated precursor intensity and computing TPR_ℓ within each bin. This reveals at which signal levels each tool begins to lose sensitivity.

1.7.4 Match-Between-Runs Error Metrics

For the Match-Between-Run (MBR) benchmark, the simulated ground truth provides both the identity of each ion and the precise boundaries of its simulated elution peak. Reported MBR-derived ions are evaluated on two independent criteria, following the framework of PIP-ECHO [**?**]. The analysis is restricted to unmodified peptides (excluding cysteines, which carry a fixed carbamidomethylation in the simulation) and to ions with isoleucine/leucine equivalence applied (I→L substitution), matching the conventions applied during simulation and search. Blank-run identifications are subtracted prior to all calculations.

Let $\mathcal{M}$ be the set of MBR-matched ions (unique sequence+charge keys) reported by the software, and let $\mathcal{G}_{\text{ion}}$ be the corresponding ground truth ion set from the simulation.

Identification error. An MBR-matched ion is an identification error if its (sequence, charge) key is absent from the simulation:

$$\text{ID Error} = \frac{|\mathcal{M} \setminus \mathcal{G}_{\text{ion}}|}{|\mathcal{M}|}. \tag{3}$$

This rate is analogous to the standard FDR of Eq. 1 but applied to MBR-derived identifications only.

Peak matching error. Separately from identification errors, a peak matching error is assigned to a correctly identified ion when its reported apex retention time $\hat{t}$ falls outside the simulated FWHM window of the peptide's elution curve. The FWHM boundaries $[t_{\text{start}}^{\text{FWHM}}, t_{\text{end}}^{\text{FWHM}}]$ are derived from the full-width at half-maximum of the simulated EMG elution profile (Eq. 8), computed from the discrete frame-abundance array stored in the simulation database. The peak matching error rate is

$$\text{Peak Matching Error} = \frac{\left|\{i \in \mathcal{M} \cap \mathcal{G}_{\text{ion}} : \hat{t}_i \notin [t^{\text{FWHM}}_{i,\text{start}},\, t^{\text{FWHM}}_{i,\text{end}}]\}\right|}{|\mathcal{M} \cap \mathcal{G}_{\text{ion}}|}. \tag{4}$$

The numerator and denominator are restricted to correctly identified ions ($\mathcal{M} \cap \mathcal{G}_{\text{ion}}$); ions that are absent from the simulation altogether contribute only to the identification error (Eq. 3) and not to this count. The two rates share the same denominator and are independent metrics. Both are expressed as percentages.

1.7.5 False Localization Rate for Phosphosite Localization

For the phosphoproteomics benchmark, the simulation assigns each phosphopeptide a definitive phosphorylation site at a specific amino acid position. Two replicates per complexity level are generated with the phospho group placed at different phosphorylatable residues (site A vs. site B), ensuring that alternative localizations are always present in the data.

A reported phosphopeptide ion (modified sequence, charge) is a *true localization* (TP) if and only if it matches an ion in the simulation ground truth $\mathcal{G}_{\text{phospho}}$, i.e. the UNIMOD:21 annotation appears at the correct amino acid position. An ion is a *false localization* (FP) if it is reported by the software but is absent from $\mathcal{G}_{\text{phospho}}$, either because the phospho mark is assigned to the wrong site within an otherwise correctly identified peptide, or because the entire precursor identification is incorrect.

The false localization rate (FLR) is evaluated as a function of the site localization probability threshold p reported by each tool. For a given threshold p, let $\mathcal{S}(p) = \{i \in \mathcal{R}_{\text{phospho}} : p_i \geq p\}$ be the set of phosphopeptide ions passing the threshold. Then

$$\begin{aligned} \text{TP}(p) &= |\mathcal{S}(p) \cap \mathcal{G}_{\text{phospho}}|, \\ \text{FP}(p) &= |\mathcal{S}(p) \setminus \mathcal{G}_{\text{phospho}}|, \\ \text{FLR}(p) &= \frac{\text{FP}(p)}{\text{TP}(p) + \text{FP}(p)}. \end{aligned} \tag{5}$$

FLR curves are generated by sweeping p from 0 to 1, producing a monotone relationship between the stringency of the site probability cutoff and the fraction of false localizations among retained identifications. This formulation is structurally identical to Eq. 1 but applied to phosphopeptide ions at the modified-sequence level, where the ground truth distinguishes not only the peptide identity but also the exact site of modification.

2 Detailed Description of the Simulation Pipeline

This section describes the workflow and key components involved in generating simulated timsTOF proteomics mass spectrometry raw data in dda-PASEF and dia-PASEF-like acquisition modes in greater detail. We define some key terms used throughout the sections:

- **tims-scan**: A tims-scan is a single ion mobility scan (effectively, one mass spectrum) where all detected signals (called events in Bruker's own nomenclature) correspond to the same ion mobility bin. Those are essentially all ions measured by TOF after one discrete push out of the second TIMS.

- **tims-frame**: A tims-frame refers to a collection of tims-scans acquired during one complete emptying of the trapped ion mobility ramp, thus spanning the full range of ion mobilities.

- **tims-cycle**: A tims-cycle consists of a full sequence of tims-frames and their associated tims-scans, typically corresponding to one complete iteration over all quadrupole settings in any PASEF acquisition mode. It includes one MS1-like precursor frame (i.e., acquired without fragmentation), followed by a series of MS2-like frames in which the quadrupole either steps through its pre-configured isolation windows (dia-PASEF), or on-the-fly scheduled selection windows, e.g., in top-N mode (dda-PASEF) before the cycle repeats.

- **simulation blueprint**: A simulation blueprint is a (partially) constructed SQLite database that defines the precursor and fragment ion space for a simulation. It enumerates all peptides and ions to be embedded in the synthetic raw data, along with their total counts (expected intensities) and distributions across retention time and ion mobility. When used in partial mode, fragment ions are not included at this stage. Once a fragmentation scheme (e.g., DDA or DIA) is applied, the blueprint is completed and can be used to generate raw datasets. This separation allows for efficient reuse: a single partial blueprint can serve as the foundation for simulating multiple acquisition strategies, enabling rapid and reproducible comparisons across different selection methods and layouts.

The `timsim` pipeline is organized into three main stages for an *in silico* run of any PASEF acquisition:

1. **Precursor space generation**. This stage includes peptide digestion and abundance sampling, calculation of retention time and charge state distributions, sampling of ionization efficiencies, calculation of ion mobility distributions, and precursor isotopic pattern generation. What is obtained are expected intensities of precursors at different stages of their *in silico* measurement. The process consist of multiplying initial peptide (or protein) intensities by consecutive probabilities of being in a different place in the physical space, in retention time, inverse ion mobility, while factoring in impact of charge. This process is mostly deterministic, except for introduction of peptide specific variability in retention time, as described in detail in sections below.

2. **Acquisition layout specification and application**. In this step, ion transmissions are determined either by simulating an online precursor selection algorithm (for dda-PASEF) or by applying a predefined isolation window scheme (for dia-PASEF). This is followed by fragment space generation, which computes fragment masses, relative intensities, and isotopic patterns for all selected precursors. Here, precursor intensities are reused and further scaled down appropriately.

3. **Raw data generation**. Once the simulation blueprint is complete, raw data is generated frame by frame in a map-reduce-like fashion. Each ion contributes signal to a number of frame and scan combinations, which are then grouped, converted into peak collections, assembled into individual tims-scans, and finally organized into tims-frames. In the process, the expected values obtained in the blueprint are further discretised to correspond to the natural frame-scan-tof grid specific to the timsTOF's raw data format.

These stages are described in greater detail in the following sections.

2.1 Precursor Space Generation

This section outlines the process by which `timsim` generates the precursor space, which is subsequently stored in a SQLite database. It results in a partial simulation blueprint, ready to simulate precursor tims-frames.

2.1.1 Protein and Peptide Generation

`timsim` can simulate datasets starting at the protein level. Users can provide a `.fasta` file containing protein sequences and specify digestion rules using an *in silico* enzyme setting, such as tryptic or unspecific digestion. By default, protein abundances are sampled from a custom "hockey-stick" distribution that combines linear and exponential decay behavior, see Eq. 6.

$$h(r) = h_0 \cdot 2^{-\delta \cdot r} \cdot 2^{-\delta_{\mathrm{exp}} \cdot \sum_{i=1}^{r} e^{-i/\mathrm{exp}_{\mathrm{red}}}} \tag{6}$$

Here, r is the protein rank, and the constants $h_0 = 10^4$, $\delta = 0.0011$, $\delta_{\mathrm{exp}} = 0.005$, and $\mathrm{exp}_{\mathrm{red}} = 600$ control the overall shape of the distribution. This form ensures a steep drop-off in abundance after a few highly abundant proteins, mimicking biological datasets where a small number of proteins dominate the signal.

Peptide abundances are then derived by applying a randomly sampled ionization efficiency factor to each peptide. This factor is drawn from the cumulative distribution function (CDF) of a normal distribution in log space, resulting in approximately exponentially decaying peptide efficiencies. Since downstream processing only requires peptide abundance annotations, users can easily customize or replace this generation scheme with alternative models.

2.1.2 Retention Time Distribution Simulation

Point estimates for expected retention times are generated by a custom deep recurrent neural network using GRU units (f_{GRU}) [**?**]. The network is trained on an established collection of indexed retention time (iRT) values [**?**] to perform the regression task of translating a given peptide's amino acid sequence into the indexed retention time of the mode of it's elution curve ($\mathsf{iRT}_{\mathsf{pep}}^{\mathsf{mode}}$), which is then linearly scaled to its actual retention time $\mathsf{RT}_{\mathsf{pep}}^{\mathsf{mode}}$, depending on the gradient length t_{gradient}, according to

$$\mathsf{RT}_{\mathsf{pep}}^{\mathsf{mode}} = \frac{\mathsf{iRT}_{\mathsf{pep}}^{\mathsf{mode}} - \min \mathsf{iRTs}}{\max \mathsf{iRTs} - \min \mathsf{iRTs}} \times t_{\mathsf{gradient}}, \tag{7}$$

where iRTs is the set of modes of all simulated peptides $\mathsf{iRTs}_{\mathsf{pep}}^{\mathsf{mode}}$, each one GRU-simulated $\mathsf{iRT}_{\mathsf{pep}}^{\mathsf{mode}} = f_{\mathsf{GRU}}(\mathsf{sequence}_{\mathsf{pep}})$.

With peptides' modes at hand, we proceed with modeling their overall shape in retention time around the expected position. To model that shape, we make use of the exponentially modified Gaussian distribution (EMG), a convolution of a Gaussian and an exponential distribution, with the density given by

$$f_{\mu,\sigma,K}(x) = \frac{1}{2\sigma K} \exp\left(\frac{1}{2\sigma K}\left(2\mu + \frac{\sigma}{K} - 2x\right)\right) \operatorname{erfc}\left(\frac{\mu + \frac{\sigma}{K} - x}{\sqrt{2}\sigma}\right). \tag{8}$$

The EMG density is commonly used as a model for chromatographic peaks [**?**, **?**, **?**, **?**, **?**] and is similar to the normal distribution. The μ and σ parameters describe the position and width of the curve while it's skewness is adjusted by K. Other EMG parametrizations make use of μ, σ and λ with $K = \frac{1}{\lambda\sigma}$. The mode of the EMG PDF can be derived by maximizing Eq. 8 with respect to x. We assume it corresponds to our GRU prediction RT, leading to an expression that allows to calculate the missing μ parameter from 8 given σ and K using

$$\mathsf{RT}^{\mathsf{mode}} = \mu - \sqrt{2}\sigma \operatorname{erfcxinv}\left(K\sqrt{\frac{2}{\pi}}\right) + \frac{\sigma}{K}. \tag{9}$$

Above, we drop the dependence on peptide for simplicity. For each peptide however, the remaining parameters σ and K are peptide specific and independently drawn. Both σ and K are drawn from a translated and scaled beta distribution, parameters of which the users can define. Thus, the user specifies the lower and upper values for σ, referred to as σ_l and σ_u, as well as the concentration parameters of the beta distribution, called σ_α and σ_β. The final simulated value for a given peptide will thus come from the following procedure

$$\begin{aligned} \hat{\sigma}_j &\sim \mathrm{Beta}\left(\sigma_\alpha, \sigma_\beta\right) \\ \sigma_j &= \sigma_l + \hat{\sigma}_j\left(\sigma_u - \sigma_l\right) \end{aligned} \tag{10}$$

In our model, the dual TIMS is considered loss-free, meaning the amount of peptides released during the recording of a single frame can be calculated by integrating the chosen distribution over the time interval between the frame start and frame end.

2.1.3 Default parameters for the simulation of retention time curves

In `timsim`, a peptide's elution curve is modeled by an exponentially modified gaussian (EMG) distribution. The EMG distribution is defined by the three parameters μ, σ and K. While μ affects its position, higher values for σ and K increase the width and skewness of the distribution, respectively. In the current implementation the position parameter μ is calculated based on the mode $\mathrm{RT}_{\mathrm{pep}}^{\mathrm{mode}}$ of a peptide's elution curve, which is estimated from the peptide's amino acid sequence by a deep learning model. The parameters σ and K, however, are not predicted based on a peptide's sequence but randomly sampled from scaled beta distributions to introduce variability into the synthetic data. Both of the scaled distributions are defined by an α and a β value for the underlying beta distribution and a lower and an upper limit of the interval upon which the beta distribution is linearly scaled. These variables are user-defined. There are default values, however, these were not derived from data but chosen solely by visual inspection (Fig. 1a). Hence, we strongly recommend replacing the default values with values that fit the experimental setup of your interest more precisely. The default values are shown in Tbl. 2.

Parameter	Description	Variable Name	Default Value
t_g	Gradient Length in seconds	‘gradient_length‘	3600
σ_l	Lower bound of EMG parameter σ	‘sigma_lower_rt‘	None
σ_u	Upper bound of EMG parameter σ	‘sigma_upper_rt‘	None
σ_α	α parameter of beta distribution scaled to (σ_l, σ_u)	‘sigma_alpha_rt‘	4
σ_β	β parameter of beta distribution scaled to (σ_l, σ_u)	‘sigma_beta_rt‘	4
K_l	Lower bound of EMG parameter K	‘k_lower_rt‘	0
K_u	Upper bound of EMG parameter K	‘k_upper_rt‘	10
K_α	α parameter of beta distribution scaled to (K_l, K_u)	‘k_alpha_rt‘	1
K_β	β parameter of beta distribution scaled to (K_l, K_u)	‘k_beta_rt‘	20

Table 2: Elution curve simulation: Overview of default parameter values.

The default values for σ_l and σ_u are 'None', as these are dynamically set depending on the gradient length t_g (see Eq. 11), if not defined by the user.

$$\begin{aligned} \sigma_u &= 1.25\sigma_m \\ \sigma_l &= 0.75\sigma_m \\ \sigma_m &= \frac{0.75 t_{\text{gradient}}}{60^2} + 1.125 \end{aligned} \tag{11}$$

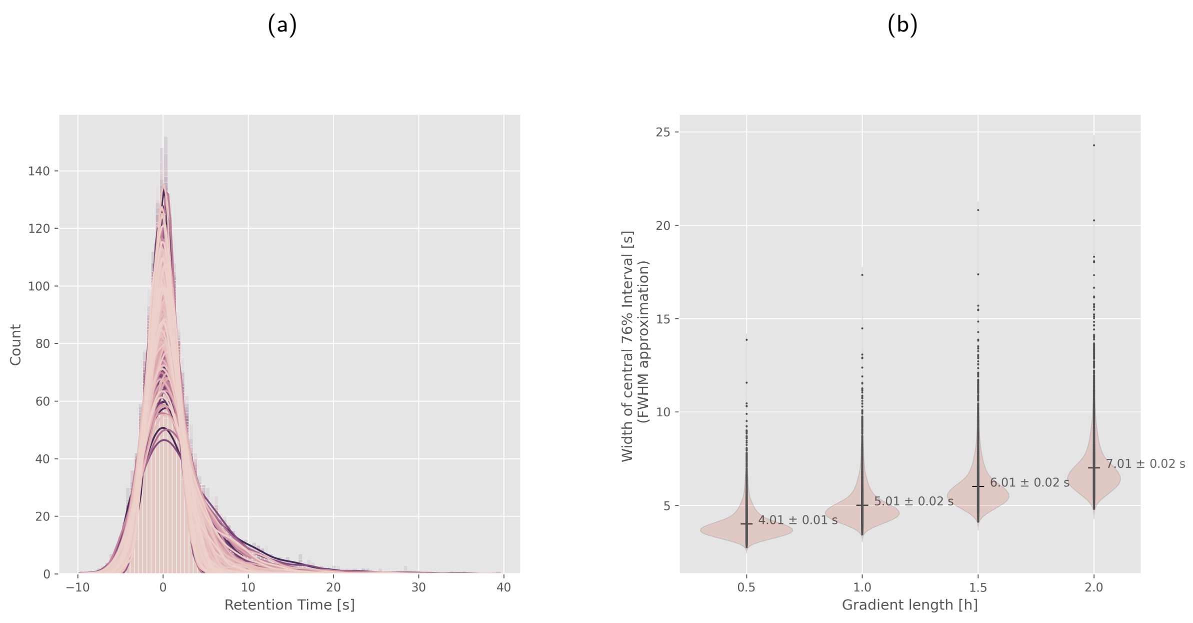

Figure 1: (a) Visual inspection of the default parameters for the generation of peptide elution curves. Shown are the histograms of 500 EMG distributions (1000 samples each) with mode zero and σ, K sampled from scaled beta distributions with the default parametrization (60 min gradient, $t_g = 3600$). (b) Violinplot of the widths of central 76% intervals of 5000 EMG distributions with sampled σ, K using the default parametrization at various gradient times ($\mu = 0$). The parameters σ_l and σ_u were calculated via Eq. 11. Annotation gives mean of the widths and its bootstrapped standard deviation (5000 bootstrap samples, 100 iterations).

Here, σ_m is the middle of the interval from which σ is sampled. The calculation of σ_m in Eq. 11 was chosen such that the mean central 76% interval width of a sample of curves is approximately 4 s for 30 min gradients and 7 s for 120 min gradients, as reported by Meier *et al.* for experiments with the timsTOF Pro device (Fig. 1b,[**?**]). The central 76% interval would correspond to the full width at half maximum (FWHM) in case of a fully gaussian curve.

2.1.4 Modelling Charge State Distribution

`timsim` provides two different models for estimating the distribution of charge states for a given peptide sequence. The first is a binomial model that was adopted from previous publications in the field [**?**, **?**]. This model relies on the number of basic amino acids (n_b), as these are together with the peptide's N-terminus considered protonizable. The protonation probability p can be set by the user. The fraction of peptide ions with charge z is then given by the probability of that charge in the binomial model (Eq. 12).

$$\mathrm{P}(z) = \binom{n_b + 1}{z} p^z \, (1 - p)^{n_b + 1 - z} \tag{12}$$

The second option is a custom deep recurrent neural network, utilizing GRU units, trained on tryptic HeLa data from a collection of timsTOF DDA experiments to return a probability distribution over the observed charge states. It is worth noting that software often finds it more challenging to identify peptide ions carrying a single charge compared to those with higher charges [**?**, **?**], which may introduce a bias into the training set of the charge state distribution estimator. Despite this potential bias, we believe it is valuable to include this approach, as it can be flexibly fine-tuned for other scenarios and datasets.

The default parameters for the binomial ionization model are listed in Tbl. 3. The default value for the protonation probability p was adopted from Bielow *et al.* [**?**].

Parameter	Description	Variable Name	Default Value
p	Protonation probability	'p_charge'	0.8
$z_{\max}$	maximum considered charge state	'max_charge'	4
$\mathrm{P}_{\min}$	minimum probability of a charge state, charge states below will be dropped	'min_charge_contrib'	0.005

Table 3: Ionization simulation: Overview of default parameter values.

With the parameters $z_{\max}$ and $\mathrm{P}_{\min}$ users can restrict which charge states will be stored in the SQLite database and hence be present in the simulated data. In Eq. 13 S_z represents the set of charge states of a peptide that are stored in the database.

$$S_z = \{z \in \mathbb{N} | \, (z \leq z_{\max}) \wedge \mathrm{P}(z) \geq \mathrm{P}_{\min}\} \tag{13}$$

$\mathrm{P}(z)$ is the binomial probability of the charge state z for a given peptide with n_b basic amino acids (Eq. 12). Together with the N-terminus, the peptide has $n_b + 1$ protonizable sites.

All charge states in S_z are stored together with their probabilities, which will later reflect the amount of peptides in these charge states. By default the stored probabilities will be normalized, hence the probabilities of all stored charge states in S_z will sum to 1.

2.1.5 Inverse Ion Mobility Distribution Calculation

`timsim` models the inverse ion mobility of each peptide ion as a normal distribution, parameterized by a mean and variance predicted by a neural network, using an updated version of the `ionmob` predictor [**?**]. For each peptide ion i, the network takes the mass-to-charge ratio $(\mathrm{m/z})$, charge state (z), and amino acid sequence $(\mathrm{sequence})$ as input, and outputs the parameters of a collision cross section (CCS) distribution:

$$\mathrm{NN}(\mathrm{m/z}_i,\ z_i,\ \mathrm{sequence}_i)\ =\ \mathcal{N}(\mu_i,\ \sigma_i^2)$$

The predicted mean μ_i and variance σ_i^2 correspond to a normal distribution over CCS values. These are subsequently converted to inverse reduced ion mobility values using the Mason–Schamp equation [**?**].

2.1.6 Isotopic Pattern Calculation

For each precursor ion (and later its fragments), peaks are generated not only as monoisotopic peaks but also as a group of isotopologues. The relative abundances of these isotopologues are determined by the isotopic compositions of the contained elements and their atomic counts. Given the known exact atomic composition of the peptides and their fragments, we use a convolution algorithm to calculate the probabilities of different isotopologues, as detailed in *Smith and Martin* [**?**].

2.2 Acquisition Layout Specification and Application

After the simulation of the precursor space, the simulation blueprint is finalized by applying a selection scheme to the precursor ions. This allows to calculate the ion transmissions and collision energies applied to a specific transmission which are necessary for the creation of fragment ions.

2.2.1 Quadrupole Transmission Calculation

At the core of any PASEF acquisition strategy is the way the quadrupole selects ions from the ion mobility-separated ion cloud for fragmentation, by isolating specific regions in the mass-to-charge (m/z) dimension. This selection process may either be performed online (as in dda-PASEF, where precursors are selected dynamically during acquisition) or offline using a predefined, periodically repeating window scheme (as in dia-PASEF).

To model the smooth transmission profile of the quadrupole, we define a differentiable window function $T_{k,m_{f,s},w_{f,s}} : \mathbb{R}_{\geq 0} \to [0,1]$, which describes the probability that an ion of mass-to-charge ratio x is transmitted through the quadrupole in frame f and scan s. The function is parameterized as follows:

- $k \in \mathbb{R}^+$ controls the steepness of the window edges (i.e., how sharp the transition is between transmitted and filtered-out ions),
- $m_{f,s} \in \mathbb{R}^+$ is the center of the transmission window in Thomson,
- $w_{f,s} \in \mathbb{R}^+$ is the full width of the transmission window in Thomson.

The function is constructed as the difference of two logistic (sigmoid) functions to give a smoothed rectangular window:

$$T(x \mid k, m_{f,s}, w_{f,s}) = \frac{1}{1+e^{-k\cdot(x-(m_{f,s}-\frac{w_{f,s}}{2}))}} - \frac{1}{1+e^{-k\cdot(x-(m_{f,s}+\frac{w_{f,s}}{2}))}} \tag{14}$$

This formulation ensures a smooth transition at the window edges, which better approximates the actual physical transmission characteristics of the quadrupole compared to idealized step functions. The smoothness parameter k can be tuned to simulate sharper or softer isolation behavior depending on the desired fidelity.

Collision Energy Function In addition to the transmission window, we offer users the possibility to use different frame-scan specific collision energies, $C(\text{frame}, \text{scan}) > 0$. In practice, they can be constant, stepped or ramped functions over the ion mobility dimension, or a calibrated curve (e.g., decreasing energy for higher mobility values) depending on the acquisition scheme used. Together, T and C define the essential scan-wise properties that govern ion selection and fragmentation behavior during synthetic acquisition.

2.2.2 DDA Mode

In real dda-PASEF experiments, the data acquisition layout is generated on-the-fly. During acquisition, a top-intensity precursor selection algorithm is executed in real time on each incoming MS1 frame, guided by a set of user-defined parameters. These typically include: (1) the maximum number of precursor ions to be selected per frame, (2) the number of MS2 (fragment) frames to follow each MS1 frame, (3) a minimum intensity threshold below which precursor ions are ignored, and (4) an exclusion duration defining how many frames a previously fragmented ion is to be skipped from re-selection. The selection algorithm typically operates in a top-N fashion, where precursor ions in a given frame are ranked by intensity and the highest-ranking ions are selected for fragmentation across the following k MS2 frames. In parallel, a feature detection step identifies key isotopic features such as the monoisotopic m/z, apex intensity, and total signal intensity of the selected precursor ion.

timsim mimics this behavior by iterating over all precursor frames in a simulation and identifying all ions present in each frame. For each frame, precursor ions are assigned an intensity based on their simulated relative abundance, sorted in descending order, and the top-N ions are scheduled for fragmentation. This scheduling process is repeated across the entire experiment, producing a complete sequence of quadrupole isolation settings that define the final raw data layout. Since isotope patterns are explicitly simulated for each ion, the feature detection step is trivially satisfied by directly extracting known properties from the simulation, resulting in a perfect (i.e., noise-free) annotation of isotopic features. While this approach ensures idealized ion selection and feature detection, an alternative mode, where actual peak picking and feature detection are applied to the simulated raw data, is not yet implemented. We plan to introduce this more realistic selection strategy in a future version of timsim.

2.2.3 DIA Mode

The dia-PASEF data acquisition layout defines how continuously arriving ions from liquid chromatography are trapped, mobility-separated, quadrupole-selected, fragmented, and recorded into mass spectra by the timsTOF instrument. It is specified offline, meaning before the acquisition of the data is performed. For readers unfamiliar with the basic concepts of PASEF, we refer to the original dia-PASEF publication [**?**]. It is possible to specify the layout of PASEF transmission windows and collision energies via configuration files or by providing a real reference dataset.

2.2.4 Fragment Ion Generation and Fragment Intensity Prediction

The rustims framework includes functionality to calculate all expected fragment m/z values for peptide ions. This process exhaustively generates all theoretically possible fragment ions, not including neutral losses. However, to achieve a more realistic distribution of ions, relative fragment intensity prediction is performed. Fragment intensity prediction is carried out using the timsTOF optimized release of the Prosit model [**?**], a deep learning model that accurately predicts the expected relative occurrences of b and y ions. Additionally, users can specify a down-sampling factor, where fragments are randomly removed with the probability of removal being inversely proportional to their intensity. For simplicity, we do assume that the fragmentation is exhaustive and there no precursor signal is present in the fragment spectra.

2.2.5 Remote Prediction via KOINA

As an alternative to the locally executed deep learning models described in the preceding sections, timsim supports remote prediction through KOINA, a community prediction service hosted at koina.wilhelmlab.org. KOINA exposes a collection of state-of-the-art proteomics prediction models via a gRPC interface, which timsim accesses through the koinapy Python client. This integration covers all three prediction tasks in the simulation pipeline: retention time, collision cross section (and thereby ion mobility), and fragment ion intensities.

For retention time prediction, the available remote models include DeepLC, Chronologer, AlphaPeptDeep, and Prosit 2019 iRT. For collision cross section and ion mobility prediction, users can select AlphaPeptDeep or IM2Deep. For fragment intensity prediction, the supported models are Prosit 2023 (timsTOF-optimized), AlphaPeptDeep, and MS2PIP (timsTOF 2023 and 2024 variants). The choice of remote model is configured in the `[models]` section of the TOML configuration file via the `rt_model`, `ccs_model`, and `intensity_model` fields; leaving these fields empty or setting them to `local` selects the default local models.

Because the available models differ in their supported input space, `timsim` applies model-specific input filters before dispatching predictions. For instance, Prosit models accept sequences of at most 30 amino acids with only carbamidomethylation on cysteine and oxidation on methionine as variable modifications, whereas AlphaPeptDeep models impose no such restriction and support the full range of modifications including phosphorylation. Sequences that fall outside a model's supported input space are excluded from the remote prediction and handled by the local fallback model. More generally, if the KOINA service is unreachable or a prediction call fails, the pipeline automatically falls back to the corresponding local PyTorch model with a logged warning, ensuring that simulations complete without manual intervention.

This integration enables users to leverage externally hosted, regularly updated prediction models without requiring local GPU resources or model weight files, and facilitates reproducibility through versioned remote model endpoints.

2.3 Raw Data Generation

Raw data is generated using the finalized blueprint in the SQLite database created in the previous step, which contains information about the occurrence and abundance of all peptide ions. Raw data can be generated in two ways: The first approach involves running a full simulation of a timsTOF raw dataset, which is then saved to disk in the TDF format. Alternatively, users can connect to the SQLite database created during the first part of the simulation. This method allows for on-the-fly generation of single tims-frames or groups of tims-frames using a Python interface, which is particularly useful when focusing on specific aspects of the simulation, such as transmission patterns for certain peptides.

2.3.1 Tims Frame Assembly

In full dataset generation mode, the tims frame assembly process is responsible for generating tims frames from the blueprint stored in the SQLite database. This operation is computationally intensive, as it requires aggregating all peptide ions contributing signals to a tims frame along with their respective intensities. However, because the generation of individual tims frame is independent of one another, this process can be parallelized. We implemented the generation in a map-reduce-like fashion, where individual ions emit their mass spectrum and contribution to all relevant frames and scans. Parallelization occurs at the frame level, where individual spectra are first merged on the scan level and then on the frame level. As a result, the overall speed of the generation process can be significantly increased by leveraging multi-core systems.

2.3.2 Addition of Noise

The addition of noise to the expected signals is categorized into two main categories. First, there is addition of noise to the generated peptide signal distribution across the retention-time, ion-mobility, and m/z dimensions, effectively distorting the smooth expected distributions. This is achieved by introducing random noise within a user-defined ppm error range to peaks in the m/z dimension, along with uniformly distributed noise in the retention time and ion mobility dimensions, with the noise strength configurable by the user. Second, there are noise peaks that cannot be attributed to any peptide ion, potentially resulting from chemical or electrical noise. Synthetically generating realistic noise peaks has long been a challenge in the *in silico* creation of MS datasets

[**?**], as it heavily depends on the specific setup and sample. Our approach differs from previous methods by allowing the use of real data signals to be mixed with the synthetic dataset. Peaks up to a user-defined threshold are randomly sampled from a given reference dataset, bypassing the need for complex noise modeling that might ultimately fall short of replicating the distributions found in real datasets. For instance, a potential use case could involve measuring either a blank or a contaminant dataset as a reference, which can then be combined with the simulated signal to create a more realistic dataset.

2.3.3 TDF Writer

We developed a writer for the timsTOF-specific TDF format. Details about the format can be found in previous works [**?**, **?**]. Briefly, timsTOF raw data is stored by the vendor in a .d folder, which contains an SQLite database holding metadata such as quadrupole settings and frame peak counts, along with a binary file where TOF, scan, and intensity values are stored as a single consecutive array of zstd[2] compressed bytes. Our writer reverses this process by translating generated m/z and IM values to TOF and scan values. This translation relies on a real reference dataset and a call to the vendor's proprietary translation functions, which are accessed through their binaries. The data is then converted into a stream of bytes, compressed with zstd, and written to disk. Additionally, our writer generates a SQLite database that contains the necessary metadata tables, using calibration parameters from the reference file.

2.3.4 On-The-Fly Generation Mode

`timsim` can operate in an on-the-fly generation mode, allowing users to connect to a ready-to-simulate layout database via Python. This mode enables the generation of not only full datasets that can be saved in vendor format but also individual frames or groups of frames online.

2.4 Current Software Limitations

At the time of writing, generating synthetic TDF files requires a real reference dataset acquired using a Bruker timsTOF instrument. This is necessary because the vendor does not disclose the conversion formulas for translating TOF indices to m/z values and scan values to inverse ion mobilities. The TDF format stores indexed TOF and scan values that are compressed on a frame-by-frame basis and are only translated back to meaningful physical values during data loading. However, saving to TDF is desirable as it allows generated datasets to be used with software such as MaxQuant, FragPipe, PEAKS, or DIA-NN. This approach currently restricts the number of scans and the ion-mobility range to the layout in the reference file, a limitation we are actively working to address with Bruker team members in a future version of `timsim`. While `timsim` is generally designed with cross-platform compatibility in mind, the reliance on platform-specific Rust binaries and PyTorch currently limits full support to Linux and macOS systems. Windows users are advised to run `timsim` via the Windows Subsystem for Linux (WSL), which provides a stable and fully supported environment.

[2]https://github.com/facebook/zstd