UNIVERSITY OF CALIFORNIA, IRVINE

Streamlining Cross-Linking Mass Spectrometry Analysis

DISSERTATION

submitted in partial satisfaction of the requirements for the degree of

MASTER OF SCIENCE

in Biomedical Sciences

by

Alex Huszagh

Dissertation Committee:
Professor Lan Huang, Chair, Chair
Professor Rongsheng Jing
Associate Professor Feng Qiao

2017
DEDICATION

I would like to dedicate this degree to my grandmother, who passed away on the day of my Master’s defense. You taught me the world was much broader than the culture I was raised in, and were an inspiration to me. Rest in peace.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF FIGURES</td>
<td>vi</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>ix</td>
</tr>
<tr>
<td>ACKNOWLEDGMENTS</td>
<td>x</td>
</tr>
<tr>
<td>CURRICULUM VITAE</td>
<td>xi</td>
</tr>
<tr>
<td>ABSTRACT OF THE DISSERTATION</td>
<td>xiii</td>
</tr>
<tr>
<td>1 Significance of Cross-Linking Mass Spectrometry</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Structural Biology</td>
<td>2</td>
</tr>
<tr>
<td>1.3 Structural Techniques</td>
<td>3</td>
</tr>
<tr>
<td>1.4 Cross-linking Products</td>
<td>9</td>
</tr>
<tr>
<td>1.5 XL-MS Workflow</td>
<td>10</td>
</tr>
<tr>
<td>1.6 Modeling</td>
<td>12</td>
</tr>
<tr>
<td>1.7 Discussion</td>
<td>13</td>
</tr>
<tr>
<td>2 Creation of an XL-MS Data Pipeline</td>
<td>15</td>
</tr>
<tr>
<td>2.1 Introduction to XL-MS Data Analysis</td>
<td>15</td>
</tr>
<tr>
<td>2.2 Approach</td>
<td>18</td>
</tr>
<tr>
<td>2.3 Design</td>
<td>21</td>
</tr>
<tr>
<td>2.3.1 Abstraction &amp; Standard Library</td>
<td>22</td>
</tr>
<tr>
<td>2.3.2 I/O Interface</td>
<td>23</td>
</tr>
<tr>
<td>2.3.3 Parser Interface</td>
<td>24</td>
</tr>
<tr>
<td>2.3.4 Data Processing</td>
<td>25</td>
</tr>
<tr>
<td>2.3.5 Cross-Link Searching</td>
<td>28</td>
</tr>
<tr>
<td>2.3.6 Spectral Quantitation</td>
<td>28</td>
</tr>
<tr>
<td>2.3.7 Data Export</td>
<td>30</td>
</tr>
<tr>
<td>2.3.8 Python Interface</td>
<td>31</td>
</tr>
<tr>
<td>2.3.9 Qt5 Application</td>
<td>31</td>
</tr>
<tr>
<td>2.4 Discussion</td>
<td>32</td>
</tr>
</tbody>
</table>
3 Algorithm Development for Cross-Linked Peptide Quantitation

3.1 Introduction to Quantitative Proteomics
3.2 MS1 Quantitation Workflow
3.3 Shape-Based Peak-Picking
3.4 Wavelet-Based Peak-Picking
3.5 Simple Peak-Picking
3.6 Peak Validation
3.7 Discussion

4 XL-MS Analysis of the 26S Proteasome

4.1 Significance of the 26S Proteasome
4.2 Experimental Design
4.3 Methods & Results
4.4 Discussion

Bibliography

A General Algorithms
A.1 Pseudocode Conventions
A.1.1 Comments
A.1.2 Operators
A.1.3 Arguments
A.1.4 Branches and Loops
A.1.5 Mathematical
A.1.6 Objects
A.1.7 Collections
A.2 Utility
A.3 Numeric
A.4 Statistics
A.5 Searching
A.6 Combinations and Permutations

B Abstraction & Standard Library
B.1 Filtering Streams

C XL-MS Data Pipeline
C.1 Constants
C.2 Models
C.2.1 Mass Spectrometry Calculations
C.3 Cross-Linked Peptide Sampling
C.3.1 Objects
C.3.2 Routines
C.4 Cross-Linked Peptide Validation
C.4.1 Objects
C.4.2 Routines
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>C.5 Cross-Linked Peptide Scoring</td>
<td>95</td>
</tr>
<tr>
<td>C.5.1 Metrics Distribution</td>
<td>95</td>
</tr>
<tr>
<td>C.5.2 Composite Score Equation</td>
<td>99</td>
</tr>
<tr>
<td>C.6 Spectral Quantitation</td>
<td>102</td>
</tr>
<tr>
<td>C.6.1 Target Ion Extraction</td>
<td>103</td>
</tr>
<tr>
<td>C.6.2 SPS Ion Extraction</td>
<td>104</td>
</tr>
<tr>
<td>D Data Export</td>
<td>106</td>
</tr>
<tr>
<td>D.1 Identification Report</td>
<td>106</td>
</tr>
<tr>
<td>D.2 Table Report</td>
<td>107</td>
</tr>
<tr>
<td>D.3 Quantitation Report</td>
<td>108</td>
</tr>
<tr>
<td>E Qt5 Application</td>
<td>109</td>
</tr>
<tr>
<td>E.1 Splash Menu</td>
<td>110</td>
</tr>
<tr>
<td>E.2 Workflow Menu</td>
<td>111</td>
</tr>
<tr>
<td>E.3 Input File Table</td>
<td>112</td>
</tr>
<tr>
<td>E.4 Modification Selection</td>
<td>113</td>
</tr>
<tr>
<td>E.5 Settings Dialog</td>
<td>114</td>
</tr>
<tr>
<td>E.6 Cross-Link Viewer</td>
<td>115</td>
</tr>
<tr>
<td>F Peak-Picking</td>
<td>116</td>
</tr>
<tr>
<td>F.1 Clean Chromatograms</td>
<td>117</td>
</tr>
<tr>
<td>F.2 Noisy Baseline</td>
<td>118</td>
</tr>
<tr>
<td>F.3 Split Elution</td>
<td>119</td>
</tr>
<tr>
<td>F.4 Peak Validation</td>
<td>119</td>
</tr>
<tr>
<td>G 26S Proteasome Analysis</td>
<td>121</td>
</tr>
<tr>
<td>G.1 Replicate Reproducibility</td>
<td>121</td>
</tr>
<tr>
<td>G.2 Change in Cross-Link Abundance</td>
<td>124</td>
</tr>
<tr>
<td>G.3 Cross-Link Mapping to Reference Structures</td>
<td>125</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Page</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Diffraction pattern from a protein crystal. Diffraction of a lysozyme crystal. Diffraction patterns at many different angles are collected to reconstruct the electron density map. Adopted from Del45 (Public Domain).</td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>1.2</td>
<td>HSQC-NMR of a protein fragment. Heteronuclear single quantum coherence spectroscopy (HSQC) plots the PPM shifts of each N-H bond, one of the initial steps required structural modeling via protein NMR. Adopted from Wu et al. [1].</td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>1.3</td>
<td>Electron density map of a eukaryotic ribosome. Map of the translational initiation complex acquired via CryoEM. The projection was generated by EM Data Bank [2] for the EMD-2421 entry [3].</td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>1.4</td>
<td>Experimental and theoretical SAXS profile from Nup133. The scattering vector (q) plotted against the intensity. The theoretical profile was calculated using the FoXS module from the Integrative Modeling Platform (IMP) using data adopted from the IMP tutorial [4].</td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>1.5</td>
<td>FRET between a donor and acceptor pair. After absorption of a blue photon by the donor, the donor transfers this energy to the acceptor via FRET, which is emitted as a green photon.</td>
</tr>
<tr>
<td>8</td>
<td></td>
</tr>
<tr>
<td>1.6</td>
<td>Linkage Map of the 26S Proteasome. Inter- and intrasubunit cross-links within the 26S Proteasome. Using the cross-links as distance restraints guides the modeling of a protein or complex.</td>
</tr>
<tr>
<td>9</td>
<td></td>
</tr>
<tr>
<td>1.7</td>
<td>Cross-link nomenclature. The naming scheme and linkage schematics for cross-linking reaction products.</td>
</tr>
<tr>
<td>10</td>
<td></td>
</tr>
<tr>
<td>1.8</td>
<td>Sample XL-MS workflow using purified proteins. Purified proteins are cross-linked, separated via SDS-PAGE, the dimer excised from the gel, digested, and analyzed via MS, and the cross-links mapped to the protein sequence.</td>
</tr>
<tr>
<td>11</td>
<td></td>
</tr>
<tr>
<td>1.9</td>
<td>Cross-Link Distance Distribution from 14 Experiments. The theoretical Euclidean and SAS distances for cross-links from 14 independent experiments, using data compiled from XLdb [5].</td>
</tr>
<tr>
<td>13</td>
<td></td>
</tr>
<tr>
<td>2.1</td>
<td>Peptide Fragmentation Pattern. Schematic demonstrating the experimental fragmentation pattern for two, separate, proteolytic peptides.</td>
</tr>
</tbody>
</table>
2.2 Cross-linked Peptide Fragmentation Pattern.
   Schematic demonstrating the experimental fragmentation pattern for chemi-
   cally cross-linked peptides, which are color-coded by peptide sequence. . . .

2.3 MS-Cleavable Cross-Linked Peptide Identification.
   Schematic demonstrating the integration of data from 3 levels of MS acquisi-
   tion for unambiguous cross-linked peptide identification. . . . . . . . . .

2.4 In Silico Cross-Linking.
   Schematic demonstrating the stepwise procedure to generate theoretical cross-links
   from peptide identifications. First, combinations of the identified peptides are
   exhaustively sampled. Next, the number of intact cross-linkers is calculated
   for each combination iteratively, and the theoretical ion composition deter-
   mined using the theoretical cross-linker count and number of experimental
   cross-linker fragments. . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

2.5 XLTools data pipeline.
   First, we build upon a common abstraction platform for similar functionality
   on all operating systems. We then define a core, C++ library for cross-link
   identification and quantitation. Finally, this library is wrapped to both a
   graphical application and a Python library, allowing programmatic or appli-
   cation access. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

2.6 Filtering stream buffer schematic.
   Data is read in chunks from a source (a file, socket, etc.) by a stream, whose
   underlying buffer transforms the data via a callback before the data is read
   by the program.

2.7 Parser interface hierarchy.
   Each specialized parser inherits from a common abstract base class (ABC),
   enabling a single helper function to identify and parse all suitable input types.

2.8 MS acquisition methods.
   Sequential acquisition produces each child scan before another scan at the
   parent level is acquired. Asynchronous acquisition acquires up to $N$ MS2 scans
   and $K$ MS3 scans by cycling through MS2 scans and sequencing cross-link
   fragment ions. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

2.9 Relative isotope abundance by mass.
   Normalized abundance of a given isotope relative to all isotopic states for
   natural (unlabeled) peptides. Most cross-linked peptides have a neutral mass
   greater than 2,000 Da, where the monoisotopic peak stops being the dominant
   species. These abundances were calculated at intervals of 200 Da for a peptide
   consisting of average aminoacids, with an elemental composition of C4.94
   H7.76 N1.6 O1.48 S0.04 per aminoacid [6, 7]. . . . . . . . . . . . . . . . . . .

3.1 Generalized workflow for quantitative MS.
   Labeled proteins are cross-linked and digested with proteolytic enzymes, sub-
   mitted for multistage MS analysis, and relative abundance quantified by the
   intensity ratio between different populations. . . . . . . . . . . . . . . . . . .
3.2 Conversion of raw spectra to XICs.
Peaks from target m/z values are extracted over time to represent the intensity over time. In the case of multiple peaks within a desired mass tolerance, we sum the raw intensities and calculate the weighted average of the m/z values by the intensities. ...................................................... 35

3.3 Schematic for peak selection and integration of an XIC.
The area in green represents the signal from our analyte, while the area in red was the signal from a contaminant. ...................................................... 36

3.4 Iterative fit to a Gaussian function.
Fit of an experimental chromatogram using 800 iterations. ...................... 37

3.5 Weighted fit to a Gaussian function.
Fit of an experimental chromatogram using 800 iterations weighted by intensity². 38

3.6 CWT on random data.
The yellow regions represent areas of high amplitude, while represents low amplitude. The upper portion represents coarse features, while the lower portion represents fine features. Vertical streaks represent peaks, with the higher the amplitude and longer the streak signifying a more significant peak. 39

3.7 CWT fit to MS data.
The relative enrichment of yellow in the region from 2.0-2.5 on the X-axis signifies it contains the cross-link elution, consistent with manually curated data. ................................................................. 40

3.8 AB3D algorithm.
The algorithm finds each local maximum, and the peak elution window is identified by finding the first local minima below a certain intensity threshold. Here, the intensity threshold is set to 50% (a) or 30% (b) of the local maximum. Adopted from Aoshima et al. (Public Domain) [8]. ................. 41

3.9 Tree classifier schematic.
Simplified schematic demonstrating the simplified classification of spectra based on spectral parameters. ................................................................. 42

3.10 Scoring visualization.
Visualization of how to calculate the predicted and error area for Equation 3.1. 43

4.1 Experimental design for 26S proteasome analysis.
Cells were grown in replicate, with one population labeled and the other population unlabeled. The 26S was affinity purified and cross-linked prior to mixing, and submitted for LC-MS analysis. Candidate cross-links that were changed between populations are visualized using a volcano plot. ................. 47

4.2 Volcano plot of Pre10 linkages.
Volcano plots for the fold change in the L/H cross-link abundance. ............ 48

4.3 Correlation between proteasome structures.
The RMSD between various models of the 26S proteasome. Two models of 5LN3 were used, one denoting a structure provided during initial model generation (initial) and one corresponding to the published structure (final). 51
LIST OF TABLES

2.1 Ambiguous aminoacid placeholders and their replacements. . . . . . . . . . . 27

4.1 Proteasome substructures.
   Substructures used to normalize cross-link abundance by protein abundance. 49
ACKNOWLEDGMENTS

I would like to thank my mentor, Lan Huang, who was continually supportive of my work even as it drifted from the core focus of the lab and allowed me to take time off during challenging times. I would also like to thank my family, especially my brother, who helped me realize the significance of my work and achievements.
CURRICULUM VITAE

Alex Huszagh

EDUCATION

Master of Science in Biomedical Sciences 2017
University of California, Irvine

Bachelor of Arts in Chemistry 2013
Macalester College

RESEARCH EXPERIENCE

Graduate Research Assistant 2013–2017
University of California, Irvine

Undergraduate Research Assistant 2012
University of California, Santa Barbara

Undergraduate Research Assistant 2012
Université libre de Bruxelles

TEACHING EXPERIENCE

Teaching Assistant 2015
University of California, Irvine

Science Tutor 2011-2013
Macalester College
REFEREED JOURNAL PUBLICATIONS

Molecular Details Underlying Dynamic Structures and Regulation of the Human 26S Proteasome. 2017
Molecular and Cellular Proteomics

Partial Agonist and Biased Signaling Properties of the Synthetic Enantiomers J113863/UCB35625 at Chemokine Receptors CCR2 and CCR5. 2017
Journal of Biological Chemistry

Analytical Chemistry

Developing an Acidic Residue Reactive and Sulfoxide-Containing MS-Cleavable Homobifunctional Cross-Linker for Probing Protein-Protein Interactions. 2016
Analytical Chemistry

Biased signaling at chemokine receptors. 2015
Journal of Biological Chemistry

SOFTWARE

PyCPP https://github.com/Alexhuszagh/pycpp
Lightweight, Python-like C++ Platform Abstraction Library.

AutoCOM https://github.com/Alexhuszagh/autocom
Cross-compiler Component Object Library (COM) automation.

Breeze Style Sheets https://github.com/Alexhuszagh/BreezeStyleSheets
KDE breeze and breeze dark-like stylesheets for Qt5 applications.
Cross-linking mass spectrometry maps the structural topology of protein complexes by using the distance between linked residues as spatial constraints, complementing other structural biology techniques. However, the identification of cross-linked peptides scales poorly with the number of proteins analyzed. Our lab has previously developed MS-cleavable cross-linkers to enable the separation of cross-linked peptides prior to sequencing, enabling peptide identification using standard peptide search databases. We describe the design and implementation of platform and application named XLTools for the automated identification of MS-cleavable cross-linked peptides. XLTools supports open and proprietary data formats and common peptide search databases, facilitating its integration into existing workflows. Furthermore, we developed peak-picking and validation algorithms to enable the accurate quantitation of cross-linked peptides in complex samples. We demonstrate the application of XLTools to the quantitative analysis of the 26S proteasome cross-linked in vivo and in vitro.
Chapter 1

Significance of Cross-Linking Mass Spectrometry

1.1 Introduction

Proteins are the key effectors of cellular function, and a protein’s function is dependent on its biomolecular structure and interactions. Structural determination techniques therefore provide insight into the molecular mechanism of proteins, facilitating targeted drug discovery [9]. Although traditional protein structural determination techniques such as X-ray crystallography can resolve relatively static views of a protein \textit{in vitro} [10], they fail to capture the dynamic view of these proteins \textit{in vivo}.

Chemical cross-linking mass spectrometry (XL-MS) complements the static of traditional approaches via the identification of neighboring aminoacids \textit{in vivo} [11]. Proteins are cross-linked at neighboring lysines, the proteins extracted and digested via proteolytic enzymes, and the cross-linked residues identified via liquid chromatography mass spectrometry (LC-MS) and peptide identification. Using the linkages as spatial constraints on the topol-
ogy of the protein structure with theoretical modeling and existing structural information, cross-linking mass spectrometry can guide the structural modeling of proteins, generating low-to-medium resolution structures. Combining cross-linking mass spectrometry with differential analysis, distinct states of the protein can be identified and modeled, correlating structural changes to biological function. In recent years, XL-MS data has guided the structural modeling of macromolecular protein complexes [12–14], and elucidated how neddylation actives cullin-RING E3 ligases [15], demonstrating the complementary of XL-MS with other structural biology approaches.

1.2 Structural Biology

Biomolecular structures of proteins decode mechanistic details by identifying structural motifs necessary for biological function, complementing protein sequence analysis and functional characterization [16]. Time-resolved structural analysis identifies dynamic rearrangements required for protein activation or function. For example, the ribosome translates messenger RNA (mRNA) into aminoacid sequences, which occurs via the translocation of mRNA within the ribosome catalyzed by elongation factor G (EF-G) in prokaryotes. Structural reconstructions of the 70S ribosome in various states of protein synthesis demonstrated that EF-G acts as a Brownian ratchet, stabilizing conformations that enable translocation, confirming the prior hypothesis that EF-G acts as a switch and not an active motor [17, 18]. Structural biology therefore complements traditional functional assays to elucidate the molecular mechanism of proteins and complexes.

Likewise, the structural conservation of proteins with disparate primary sequences demonstrates the utility of structural biology for the classification of uncharacterized proteins. Gene duplication and neofunctionalization, where a paralogous copy of a gene is created and adopts a novel function, highlights the selective pressure for structural motifs
during divergent evolution. Under specific conditions, the paralogs retain low sequence similarity, however, specific structural motifs are highly conserved between them [19–21]. In the case of the major facilitator protein superfamily, high-resolution structures and homology modeling show strong tertiary structure conservation between members of the family, despite poor sequence homology [22]. Molecular mimicry, where proteins with dissimilar primary sequences evolve convergently to bind to similar interactors, further demonstrates how biomolecular structure provides functional insight. Most notably, many viral proteins mimic host structures with little to no sequence similarity [23], demonstrating the convergent evolution of structural motifs from heterogeneous sequences. Structural biology is therefore crucial for biochemical and function protein characterization.

Although biomolecular structure modeling traditionally required heterologous expression systems to acquire large quantities of protein in high purity, modern advances in structural biology enable the analysis of native proteins [4, 12, 16, 19, 24], elucidating the molecular mechanisms of proteins within their biological context. While heterologously-expressed proteins provide an initial template for derive mechanistic insight, they lack the compositional and conformational changes resulting from protein regulation, producing only a single snapshot of a dynamic system. The rapid development of novel structural techniques to study native proteins in vitro and in vivo complements existing methods, providing a more complete picture of protein structure and function.

1.3 Structural Techniques

Various structural techniques have unique advantages and disadvantages, providing complementary information from the vantage of a structural biologist. Integrating disparate data collected from multiple techniques therefore enhances the confidence and resolution of
the resulting structure, and can provide information on protein dynamics and conformational substates [4, 12].

XRD, where a target protein is crystallized into a lattice and diffracted using a beam from a synchrotron, produces a diffraction pattern characteristic of the protein structure [25]. Molecular replacement and iterative model building by fitting the protein sequence into the electron density map generates a molecular model from diffraction data [26]. X-ray diffraction scales well for static macromolecules regardless of size, comprising nearly 90% of high-resolution structural models available today [16]. However, dynamic, flexible, or unstructured regions interfere with protein crystal formation, requiring removal of these regions prior to crystallization [27]. Furthermore, structural or compositional heterogeneity introduces inaccuracies in the resulting model or completely prevents model generation [16, 27]. Finally, due to the large quantities of homogeneous protein required for crystallization, XRD requires heterologous protein expression systems, limiting the analysis of dynamically regulated systems via XRD. XRD consequently provides a snapshot of a protein divorced from its biological context.

Figure 1.1: Diffraction pattern from a protein crystal. Diffraction of a lysozyme crystal. Diffraction patterns at many different angles are collected to reconstruct the electron density map. Adopted from Del45 (Public Domain).
NMR identifies shifts in nuclear properties from spatially-proximal atoms, integrating the relative position of each aminoacid to deduce the biomolecular structure [28]. NMR can be conducted both in-solution, and with microcrystals (using solid-state NMR) [28, 29]. NMR therefore enables the analysis of flexible or unstructured proteins, and is ideally suited to the study of protein dynamics [16, 28]. However, NMR scales poorly with protein size, requiring isotopic labeling for proteins above 25 kDa, with a practical limit of 60-200+ kDa [28, 30]. NMR also requires high sample homogeneity [29, 30], restricting it to the analysis of protein structure and dynamics for small, heterologously-expressed proteins.

Figure 1.2: HSQC-NMR of a protein fragment. Heteronuclear single quantum coherence spectroscopy (HSQC) plots the PPM shifts of each N-H bond, one of the initial steps required structural modeling via protein NMR. Adopted from Wu et al. [1].

CryoEM images singular, noncrystalline macromolecules, creating high-resolution electron density maps of the protein or complex from minimal starting material [31–33]. Although originally limited to low-resolution snapshots of large, highly symmetric complexes, advances in electron detection and particle selection have generated structures comparable to those from XRD [31]. Using image processing to identify structurally similar particles compensates for sample purity, allowing atomic model generation from a discrete structure comprising a small percentage of the total sample [31, 33]. However, flexible or disordered regions can lead to image blurring through averaging [34], lowering the resolution of the resulting model. In addition, many existing structures overestimate their resolution due to
over-fitting [35, 36], and particle selection may bias results towards a template model [37]. The use of complementary structural techniques may consequently help corroborate cryoEM structures, by contributing independent, empirical validation of the particle’s model.

Figure 1.3: Electron density map of a eukaryotic ribosome. Map of the translational initiation complex acquired via CryoEM. The projection was generated by EM Data Bank [2] for the EMD-2421 entry [3].

SAXS data encodes the 1D pairwise distances between neighboring atoms, providing a topological map of the protein or complex [38]. SAXS data complements XRD and NMR model building and refinement, by providing a long-range distance constraints of atoms within the model [38]. By analyzing the radially averaged scattering pattern of proteins in solution, SAXS can quantify the contribution of a limited number of conformations to a given profile [39]. SAXS profiles can also generate low- to medium-resolution structures with *ab initio* modeling and clustering [40]. However, SAXS profiles may not be unique for a structure, preventing the generation of a single model from SAXS alone [38]. In addition, SAXS relies on high sample purity for structural modeling [38], limiting its applicability to heterogeneous or dynamically-regulated proteins.
FRET measures the efficiency of resonant energy transfer between a fluorophore donor to acceptor via acceptor fluorescence, and this efficiency correlates to the Euclidean distance the donor-acceptor pair [41, 42]. FRET allows single-molecule and aggregate analysis [41, 42], enabling real-time tracking of protein dynamics from single proteins or in vivo in response to cellular stimuli. In spite of FRET’s temporal resolution, FRET has limited spatial resolution, and can typically only detect changes from $\sim 25-140 \text{ Å}$ [43]. Additionally, FRET requires the genetic insertion of the donor-acceptor pair into the analyzed protein, requiring separate constructs per spatial constraint desired. FRET’s capacity to study real-time structural changes in proteins and complexes in vivo is consequently limited since FRET is low-throughput and has poor spatial resolution.
Figure 1.5: FRET between a donor and acceptor pair. After absorption of a blue photon by the donor, the donor transfers this energy to the acceptor via FRET, which is emitted as a green photon.

XL-MS links adjacent aminoacids for subsequent MS analysis, mapping the structural topology of a protein or complex. Due to MS’s insensitivity to sample heterogeneity, XL-MS robustly identifies proximal residues from a target protein *in vitro* or *in vivo* [12, 15, 44]. By freezing interactions and analyzing the ensemble, XL-MS also maps and identifies structural dynamic and disorder regions [12], complementing high-resolution approaches like XRD and cryoEM. Recent advances have further extended the dynamic range of XL-MS, demonstrating the unbiased structural mapping of prokaryotic and eukaryotic proteomes *in vivo* [45, 46]. Cross-link identification requires no site-specific tags and supports multiplexing [47, 48], enabling the high-throughput, comparative analysis of the structural topology of proteins. However, each linkage only imposes a probabilistic upper limit on the Euclidean distance between two residues, and therefore has poor spatial resolution [49]. Moreover, differences in the relative abundance of a linkage between two populations may not directly translate to changes in the distance separating the residues: local folding or unfolding of secondary structural motifs may modulate the likelihood of linkage formation [50]. Despite XL-MS’s poor spatial resolution, the spatial constraints derived from the cross-links satisfy a small subset of macromolecular conformations, driving hybrid modeling or refining existing structural models.
1.4 Cross-linking Products

Cross-linked peptide mixtures contain three types of cross-linking products: inter-, intra-, and deadend links. An interlinked peptide, or interlink, is multiple peptides cross-linked together, demonstrating the spatial proximity of the cross-linked residues on the macro-molecular surface [51]. Two classes of interlinks exist: inter- and intraprotein, which denote whether multiple proteins or a single protein contributes to the linked peptides, respectively [13]. Intraprotein linkages help refine a protein’s structure and validate the accuracy of a molecular model, while interprotein linkages map the subunit configuration within a complex [12, 13]. An intralinked peptide, or intralink, is a single peptide modified by a cross-linker at multiple residues, suggesting the region spanned by these residues is solvent-exposed [52]. A deadend-linked peptide, or deadend, is a peptide linked to a single cross-linker reactive site, with any other end(s) modified by the bulk solvent, showing the surface-accessibility of the residue [52]. Despite interlinks being the primary focusing of most XL-MS workflows, the mixture of products from chemical cross-linking enriches structural modeling by providing diverse data on likely solvent-exposed residues in addition to topological spatial constraints.
Multiple cross-linkers may modify a single proteolytic peptide, producing hybrid cross-linked products. Due to the high stoichiometry of cross-linker reagents to the analyzed protein, proteins are cross-linked at multiple sites. Consequently, proteolytic digestion of the target protein may result in peptides containing multiple cross-linkers, potentially linking additional peptides together. We therefore refer to cross-links by the greatest linkage type identified, where interlink > intralink > deadend. Cross-links bridging 3 or more peptides are a subset of the more general interlink, and are specifically referred to multilinks.

### 1.5 XL-MS Workflow

In the canonical XL-MS workflow, proteins are cross-linked with a lysine-reactive, homobifunctional cross-linker, digested with proteolytic enzymes into peptides, and the resulting proteolytic peptides separated via liquid chromatography (LC). The eluate from the LC separation is ionized on-line and injected into a mass analyzer for multistage MS analysis [53] (Figure 1.8). In the first stage, the mass analyzer determines the mass of the intact cross-linked peptide ion, which is subsequently cleaved in multiple steps into fragment ions corresponding to the cross-linked peptide sequence via collision-induced dissociation (CID). The mass of the intact cross-linked peptide ion and unique fragmentation pattern enable
database identification of the peptide combination, and therefore the detection of spatially proximal residues under native biological conditions.

However, the combination of peptides contributing to a linkage leads to rapid growth in search complexity, prohibiting the efficient and unambiguous identification of cross-linked peptides [51]. Restricting the database to peptide pairs within a desired mass range based on sequencing heuristics limits the theoretical time complexity [54]. However, even with optimal filtering, the number of analyzable proteins falls well short of most proteomes, limiting XL-MS’s application to highly complex samples, such as cell lysates [54]. MS-cleavable cross-linkers provide an alternate approach, by separating the cross-linked peptides via CID prior to sequencing, enabling identification via standard peptide search databases [55]. Consequently, MS-cleavable cross-linkers permit database identification in $O(n)$ time, relative to $O(n^k)$ time, where $n$ is the number of peptide chains and $k$ is the maximum number of linked peptides.

Figure 1.8: Sample XL-MS workflow using purified proteins. Purified proteins are cross-linked, separated via SDS-PAGE, the dimer excised from the gel, digested, and analyzed via MS, and the cross-links mapped to the protein sequence.
1.6 Modeling

After acquiring experimental data describing the topology of a protein complex, we can model the complex via a hybrid, step-wise approach. First, models of each subunit are acquired at a desired granularity from individual atoms to beads representing collections of aminoacids [56]. These subunit models are either generated from empirical structural data, homology modeling, or *ab initio* modeling guided by structural restraints [5, 56]. Next, the subunits are iteratively docked (using a mix of rigid and flexible docking), guided by a scoring function, to select the optimal subunit conformation(s) describing the complex [5, 56]. This scoring function is derived from the fit to a theoretical complex topology (SAXS, cryoEM), or a series of linear, Euclidean spatial constraints limiting the subunit organization within the complex (XL-MS) [5, 56]. We can further refine the generated models using the solvent accessible surface (SAS) distance as a post-filtering condition [5]. After selecting various optimal complex configurations, the structure may be locally refined to optimize fit to empirical constraints. Clustering the best scoring models detects the accuracy and the resolution of the complex model, providing empirical validation of the theoretical model.

Cross-linking data provides multiple layers of data for structural modeling. At the most rudimentary, each cross-link represents a likely upper bound (30 Å) on the distance between two aminoacids [5]. We can therefore optimize the arrangement of aminoacids or subunits within a protein or complex, respectively, by the model that violates the fewest Euclidean spatial restraints [5, 56]. Since cross-links must be solvent-accessible, we can further optimize the model using the surface-occupied distance as a spatial restraint [5]. However, the theoretical frequency of a cross-link is not constant below the upper bound, and therefore is better modeled by a harmonic function (Equation 1.1) than a binary condition [5, 56]. Cross-link pairs encoded a probabilistic, surface-accessible distance pair to accurately
guide structural modeling.

\[
f(dist) = \left( \frac{dist - x0 - \text{tolerance}}{\sigma} \right)^2
\] (1.1)

Figure 1.9: Cross-Link Distance Distribution from 14 Experiments. The theoretical Euclidean and SAS distances for cross-links from 14 independent experiments, using data compiled from XLdb [5].

1.7 Discussion

Despite the success of XL-MS to guide ab initio structural modeling, the identification and quantitation of cross-linked peptides remains challenging. In order to retain physiological conformations of protein complexes, proteins must be cross-linked at low stoichiometry, resulting in cross-linked peptides orders of magnitude lower in abundance than the proteolytic matrix [57]. Furthermore, the search space for database identification of cross-linked peptides traditionally increases combinatorially relative to traditional peptide identification [51], lowering the accuracy and confidence of cross-link identification. Although MS-cleavable cross-linkers limit this database expansion, they add an extra stage of MS analysis, prac-
tically further diluting analyte abundance. We therefore describe the design, development, and application software technologies to enhance and streamline cross-linked peptide identification and quantitation.
Chapter 2

Creation of an XL-MS Data Pipeline

2.1 Introduction to XL-MS Data Analysis

During tandem MS (MS/MS) analysis of standard proteins using bottom-up proteomics, a peptide ion is isolated and identified via its fragmentation pattern. First, a candidate ion is isolated within a mass-to-charge ratio (m/z) window [58]. The isolated ion is then cleaved at high-energy via CID, producing a fragmentation pattern characteristic of the peptide’s aminoacid sequence and its post-translational modifications (PTMs) [58]. Matching this experimental pattern to a theoretical fragmentation identifies the peptide and the quality of identification [58, 59]. Comparison to a concatenated, decoy database assesses the false positive rate, or false discovery rate (FDR), of the resulting identification [59]. MS analysis is high-throughput, exceeding 600 peptide identifications per minute [60]. MS-based proteomics therefore enables the rapid characterization of proteins in heterogeneous samples via the identification of proteolytic peptides.
The identification of cross-linked peptides using non MS-cleavable cross-linkers extends upon an MS/MS workflow, using specialized search databases for spectral matching. During the CID cleavage of a cross-linked peptide, backbone fragmentation occurs at both linked peptides, convoluting the fragmentation pattern [54]. Therefore, sequencing ions contain fragments from either peptide and the joined peptides, requiring simultaneous identification of the linked peptides [54]. Since identical peptides can be linked during protein oligomerization, the time complexity to identify a candidate ion therefore scales as $n^p$, where $n$ is the number of candidate peptides and $p$ is the maximum number of peptides contributing to a cross-linked peptide. Due to the increasing time complexity as $p$ increases, most cross-linked peptide search engines therefore consider only combinations of 2 peptides [45, 54]. Even when limiting the database to binary interactions, only $\sim$5000-10,000 proteins can be analyzed in a given experiment before search times become prohibitive [54], limiting the application of XL-MS to the analysis of biological samples. Although the identification of non MS-cleavable cross-linked peptides benefits from existing experimental workflows, it suffers from convoluted spectra and prohibitive search times, restricting its application to complex samples.
Figure 2.2: Cross-linked Peptide Fragmentation Pattern.
Schematic demonstrating the experimental fragmentation pattern for chemically
cross-linked peptides, which are color-coded by peptide sequence.

MS-cleavable cross-linkers pragmatically limit database identification to linear time, fa-
cilitating proteome-wide analysis: however, no robust tool exists to integrate data from each
stage of MS analysis to automate cross-linked peptide identification. The first stage of MS
analysis (MS$^1$) acquires the mass of the intact cross-linked peptide ion [57]. This ion is subse-
quently isolated within an m/z window and the cross-linker spacer cleaved using low-energy
CID, separating the linked peptides [57]. Each peptide is then independently isolated and
sequenced, unambiguously describing the cross-linked peptide species. The cross-linker rem-
nants, generated during peptide fragmentation, are well-characterized and can be considered
as variable modifications during peptide identification [57]. We therefore describe the de-
sign and implementation of XLTools, a C++/Python library and GUI application for the
identification of MS-cleavable cross-linked peptides, interfacing with well-known proteomics
tools and standards such as mzML, pepXML, Protein Prospector, and Mascot, as well as
proprietary vendors formats.
Figure 2.3: MS-Cleavable Cross-Linked Peptide Identification.
Schematic demonstrating the integration of data from 3 levels of MS acquisition
for unambiguous cross-linked peptide identification.

2.2 Approach

Chemical cross-linking can be simulated *in silico* via the amalgamation of theoretical proteolytic peptides and cross-linkers. During the analysis of MS-cleavable cross-linked peptides, per MS2 scan, the top \( n \) candidate ions are selected for MS3 sequencing, of which a subset \( K \) are successfully matched to peptide identifications. Sampling combinations of \( K \) and replacing cross-linker fragments with intact cross-linkers reconstructs a theoretical cross-linked peptide. Theoretical cross-links with a high degree of similarity to the experimental data for all stages of MS analysis robustly identifies the cross-linked peptide.
Figure 2.4: *In Silico* Cross-Linking.
Schematic demonstrating the stepwise procedure to generate theoretical cross-links from peptide identifications. First, combinations of the identified peptides are exhaustively sampled. Next, the number of intact cross-linkers is calculated for each combination iteratively, and the theoretical ion composition determined using the theoretical cross-linker count and number of experimental cross-linker fragments.

Due to the low computational cost of MS-cleavable cross-linked peptide identification, we chose to implement generalized algorithms that identify non-binary protein-protein interactions, increasing the number of cross-linked peptide identifications per experiment. Although the predominant species of cross-linking reactions is two linked peptides modified by a single cross-linker, various byproducts also encode structural information (Section 1.4). To avoid omitting structurally relevant data, the identification of cross-linked peptides in XLTools is agnostic to the reaction product or cross-linker. We therefore describe our approach to sample and identify candidate cross-linked peptides, since they differ significantly from existing algorithms.

We employ a brute-force, combinatorial algorithm for peptide sampling, to identify candidate peptides for *in silico* cross-linking. Since linkages containing multiple peptides of the same sequence are redundantly sequenced in existing workflows [61, 62], we decided to sample combinations of the matched ion set $K$ without replacement (Algorithm \text{PEPTIDE-SAMPLING}(K) in C.3). Therefore, the time complexity of peptide sampling scales as the summation of the
binomial coefficient over desired cross-linker products (Algorithm BINOMAL-SUMMATION\(n, i_0 = 1, i_n = n\) in A.6). Limiting the analysis to combinations of up to 2 peptides produces asymptotic run-times and output similar to existing algorithms [61], demonstrating the advantage of a generalized algorithm.

To reconstruct a theoretical cross-linked peptide and complete our brute-force sampling, we need to calculate the cross-linker count \(c\) for each peptide subset. First, \(c\) cross-linkers must be able to bridge all peptides within the set (Equation C.1). In addition, there must be at least as many cross-linker reactive sites as remnants (Equation C.2). Finally, we cannot have more intact cross-linkers than remnants (Equation C.3). Combining the maximum and minimum bounds calculates all possible cross-linker counts for each peptide subset, and therefore allows enumeration over all theoretical cross-linking products.

We then classify each candidate cross-linked peptide by matching its theoretical properties to the experimental data and select the best identification via a scoring function. First, we filter the candidate cross-link list for identifications within a desired mass tolerance and with the same charge as the experimental ion. We subsequently refine these candidates using peptide identification metrics, such as ion score and expectation value, to ensure each metric is above a base threshold. Finally, we developed a composite, weighted scoring function to rank each cross-link. We fit the distribution of ion scores, expectation values, and PPMs to theoretical functions (Subsection C.5.1). We then transformed each metric by the inverse function, applied a weight to the resulting value, and summed all values to create a linear, composite function to measure cross-link quality (Algorithm COMPOSITE-SCORE\(\text{crosslink}\) in C.4). We applied this function to rank each cross-link candidate, enabling us to select the best cross-linked peptide per scan.

We made two small adjustments to our composite scoring algorithm to make it a more robust regression. Since the expectation value is dependent on the number of peptides in the search database, we applied a square root transformation to the \(-log_{10}(ev)\)
to decrease the weight of highly significant identifications while keeping a continuous distribution (Algorithm Ev-Value(crosslink) in C.5). In addition, because previous studies demonstrate that the worst peptide identification determines cross-link quality [63], we weighted the worst peptide ion score 3x higher than the remaining ion scores (Algorithm Ion-Score-Value(crosslink) in C.5).

2.3 Design

XLTools makes extensive use interfaces using abstract base classes to create an extensible, cross-platform toolset. We developed a custom, abstraction platform on top of C++11 that provides Python-like high-level functionality, removing all OS-specific logic from XLTools (Section 2.3.1). This abstraction platform is then used to define filtering stream buffers and interfaces to process open and proprietary data formats (Section 2.3.2). We use this I/O interface to then define parsers for mass spectrometry file formats (Section 2.3.3). After normalizing and processing the parsed data (Section 2.3.4), we describe our workflow to identify cross-linked peptides (Section 2.3.5) and quantify the the cross-link abundance (Section 2.3.6). Finally, we give a brief introduction into data exportation (Section 2.3.7), our Python wrappers to facilitate custom workflows (Section 2.3.8), and our Qt5 application for end-users (Section 2.3.9).
First, we build upon a common abstraction platform for similar functionality on all operating systems. We then define a core, C++ library for cross-link identification and quantitation. Finally, this library is wrapped to both a graphical application and a Python library, allowing programmatic or application access.

### 2.3.1 Abstraction & Standard Library

In order to target multiple operating systems, we developed a lightweight, modern platform abstraction layer to avoid introducing platform-specific code in XLTools [64]. We chose to target the C++11 standard, since C++11-compliant compilers have become widespread while facilitating code readability. PyCPP uses macros to detect the system compiler, architecture, byte-order, processor, and operating system features, and implements POSIX-like routines missing among individual platforms. Although similar abstraction libraries such as Boost exist [65], the onerous compile times from Boost’s large headers made designing a lightweight alternative advantageous for rapid development.
In addition to the low-level abstraction platform, PyCPP provides numerous Python-like APIs to create a modern, high-level standard library. Since Python emphasizes readability and previous versions of XLTools used Python as its native language, Python’s standard library was an ideal template for standard library design. PyCPP includes support for regular expressions, Unicode conversion, string manipulation, HTTP requests, base-N decoding and encoding, iteration routines, and JSON and XML readers and writers, in addition to high-level path and filesystem utilities. PyCPP also features modular design and lean headers, with each module dependent on a small subset of the entire library, facilitating fast compile times and code readability.

2.3.2 I/O Interface

Modern applications handle a diverse set of input and output formats, character set encodings, and data compression. In C++, I/O transformations are idiomatically handled through filtering streams, extending streams and their underlying buffers to convert data on-the-fly. Streams are hierarchical data types that manage to read or write data from an abstract source and specialized via inheritance and virtual method overrides. Reading compressed files is particularly useful for mass spectrometry, to reduce the storage capacity required to store large datasets without a separate decompression step. Coupling file-type detection to specialized filtering streams enables reading and writing to diverse media and formats without added complexity. Filtering streams consequently are essential to XLTools’s support for compressed files and internationalization.

Although numerous filtering stream buffers (streambufs) implementations exist [65, 66], they require extensive boilerplate for each stream specialization. We therefore designed our I/O interface around a generalized filtering streambuf, `filtering_streambuf`, which uses a callback to transform bytes in the underlying buffer [64]. We provide convenient general-
ized and file-based streams that use `filtering_streambuf` as an internal buffer and may be instantiated with the transforming callback, enabling stream specialization either via stream instantiation or inheritance (Section B.1).

![Diagram of filtering stream buffer](image)

Figure 2.6: Filtering stream buffer schematic. Data is read in chunks from a source (a file, socket, etc.) by a stream, whose underlying buffer transforms the data via a callback before the data is read by the program.

### 2.3.3 Parser Interface

Due to the plurality of mass spectrometry, peptide search match, and protein record file formats, we designed common interfaces to parse various open standards and proprietary file formats. Specializing an interface to parse a given format requires a single method override that fetches the next record from an abstract input, or `null` if no records remain. Each interface includes a helper function that detects the input type, and returns the correct, instantiated parser for the caller. This interface is therefore forms the central pillar of XLTools, converting disparate inputs to an identical format. Deriving each specialized parser from an input iterator facade enables the lazy evaluation of each record, providing a clean interface with minimal overhead.
2.3.4 Data Processing

The heterogeneity of the data contained within file formats requires data normalization and correction prior to analysis. Most notably, specific peptide search engines omit relevant data required for cross-link searching and can be determined using complementary sources. Mascot also does not include functionality for uncleaved modifications, or modifications that cannot occur at a proteolytic cleavage site, requiring the filtering of search hits prior to cross-link identification. In addition, we include filters for spectral processing and linking prior to analysis. Only proprietary file formats, such as Thermo’s RAW file format, include the full scan hierarchy, requiring the manual linking of parent and child scans during spectral parsing. We also implement rudimentary peak processing filters to manipulate
spectra prior to linking or quantitation, enabling on-the-fly mass cutoffs, peak centroiding, and deisotoping.

Numerous peptide search databases or export formats, such as Mascot and pepXML, do not report the peptide start position relative to the protein sequence, preventing the identification the linkage position within the protein. We therefore localize the peptide within the protein sequence by fetching proteins using a stable identifier from a local (SQLite) or remote (UniProt KB) store and find the first sequence overlap with a leading or trailing proteolytic cut site, depending on the protease cut side.

Furthermore, all open data formats for mass spectrometry data do not contain the entire spectral hierarchy, requiring scan linking prior to correlate peptide search hits to a cross-link ion. Two primary modes of MS acquisition exist: sequential and asynchronous (Figure 2.8). Linking sequential scans is simple, since each child’s scan parent must be the nearest preceding scan. We link asynchronous scans by identifying if the mass of child ion is present in the parent scan’s fragment ion list within a desired mass threshold. When the ion matches more than one parent scan, we report all candidates, since the stringent mass thresholds during cross-link mass matching should mitigate any potential inaccuracy. If a scan already has a parent, we skip scan linking and use the reported hierarchy. We apply this linking algorithm iteratively to blocks of scans originating from a single MS1 scan (including the MS1 scan), lazily generating the complete spectral hierarchy.
Sequential acquisition produces each child scan before another scan at the parent level is acquired. Asynchronous acquisition acquires up to $N$ MS2 scans and $K$ MS3 scans by cycling through MS2 scans and sequencing cross-link fragment ions.

Finally, non-curated protein sequences frequently contain ambiguous amino acids, which denote placeholders for standard amino acids (Table 2.1), and these placeholders must be replaced for high-resolution mass matching to the experimental cross-link ion. We exhaustively sample all possible replacement amino acids using a Cartesian product on the replacement amino acid sets, and report all resulting sequences within a desired mass tolerance of the experimental ion. To limit the search space, we only accept sequences with up to 1 "X", and up to 3 "B"s, "J"s, or "Z"s.

<table>
<thead>
<tr>
<th>Placeholder</th>
<th>Replacements</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>D, N</td>
</tr>
<tr>
<td>J</td>
<td>I, L</td>
</tr>
<tr>
<td>X</td>
<td>Any</td>
</tr>
<tr>
<td>Z</td>
<td>E, Q</td>
</tr>
</tbody>
</table>

Table 2.1: Ambiguous amino acid placeholders and their replacements.
2.3.5 Cross-Link Searching

After parsing and processing our experimental data, we then group peptide search hits first by MS2 scan and then by MS3 scan prior to peptide sampling. To minimize the search space size, we fold redundant identifications, which map a matched peptide sequence to multiple proteins, into a single identification until data exportation. Ambiguous identifications, which match multiple proteolytic peptides to a single ion, must be considered separately. We therefore sample combinations of scans as described in Section 2.2, and use an inner Cartesian product to exhaustively sample all groups of ambiguous identifications. Filtering for the best ambiguous identifications, or those above a certain threshold, can practically reduce the search space, by removing low quality peptide search hits.

2.3.6 Spectral Quantitation

In addition to detecting the presence of a cross-link, we frequently want quantitative information on the relative abundance of the cross-link for differential analysis between sample populations. We therefore support isotopic labeling, label-free, and isobaric labeling-based quantitation, including synchronous precursor selection (SPS). For both isotopic labeling and label-free analysis, we use profiles that define constant modifications for an isotope-labeled state to calculate the theoretical cross-link mass over all populations, extract ion chromatograms for all cross-links (Algorithm \textsc{Extract-Target-Mz}(\textit{peaklist}, \textit{target_mz}, \textit{pred} = \textsc{Sum-Peaks}) in C.6.1), identify the elution window of the cross-link ion, and calculate the peak area over the elution window. We describe the algorithms for peak selection and to elution window identification extensively in Chapter 3. For isobaric-labeling, we quantify mass tag intensity at the MS3-level, averaged between all peptides contributing to the cross-link. In the case of synchronous precursor selection (SPS), we sample all scans in the SPS file within a symmetric retention-time window, selecting scans with ions that
have a similar neutral mass and charge to the identified cross-link within deisotoping error (Algorithm \textsc{Iterate-Sps-Scans}(\textit{crosslink, spectral list, function, rt\_delta = 90, z\_delta = 1, isotope\_delta = 2, thresh = 20, units = ppm}) in C.6.2). We subsequently average the mass tag intensity over all child scans from the selected SPS scans. By supporting multiple, independent approaches for cross-link quantitation, XLTools is amenable to numerous workflows and facilitates cross-validation between results from disparate quantitation approaches.

Due to low abundance of the monoisotopic peak for high-mass peptides, which lowers the accuracy of deisotoping algorithms, we calculate the error between the theoretical and experimental cross-link mass over a range of isotopes. Above 2,000 Da, the monoisotopic peak stops being the major species and rapidly decreases in abundance at higher masses. The low abundance of the monoisotopic peak can impede deisotoping algorithms, particularly in high-noise backgrounds, preventing accurate detection of the monoisotopic peak as the base peak in the series. Calculating the mass error between the theoretical and experimental cross-link mass over a range of isotopes enables the robust identification of cross-links from poorly deisotoped ions without relaxing the error tolerance.
Figure 2.9: Relative isotope abundance by mass. Normalized abundance of a given isotope relative to all isotopic states for natural (unlabeled) peptides. Most cross-linked peptides have a neutral mass greater than 2,000 Da, where the monoisotopic peak stops being the dominant species. These abundances were calculated at intervals of 200 Da for a peptide consisting of average aminoacids, with an elemental composition of C4.94 H7.76 N1.6 O1.48 S0.04 per aminoacid [6, 7].

2.3.7 Data Export

After cross-link identification and quantitation, XLTools exports the results to Excel’s Office Open XML (OOXML) format, with various reports to accurate and succinctly summarize a sample. XLTools reports data using 3 base formats. The identification report, where each cross-link or peptide identification constitutes its own row, simplifies post-processing of the data while containing most of the relevant spectral parameters (Figure D.1). For cross-link identifications, we calculate the maximum number of peptides in a single cross-link to determine the dimensions of the report, enabling the addition of peptide identification metrics to the report. Our table report, where we list the counts of each linkage per submitted file, gives a superficial snapshot of a sample (Figure D.2). Finally, our quantitation report,
where we list the calculated intensities of each processed scan, simplifies cross-validation validation of our quantified cross-links with raw spectral data (Figure D.3).

2.3.8 Python Interface

To facilitate the extension of XLTools with custom peak-processing filters and workflows, we designed Python wrappers for our C++ API. Python is a modern scripting language with an emphasis on code readability, and has become the preferred scripting language for many scientific applications. We wrapped our parser interface, and routines for data processing, cross-link searching, spectral quantitation, and data exportation to wrap all high-level functionality. The wrappers were generated with Boost.Python [65], using automated memory management through reference-counted smart pointers. The Python wrappers for XLTools enable rapid prototyping and custom workflows, simplifying tailoring XL-MS data analysis to individual experimental methodologies.

2.3.9 Qt5 Application

In addition to our Python bindings, we also developed a cross-platform, end-user application for standard workflows. The launch menu is a splash screen with menu buttons to select one of two predefined workflows, or one of two document editors (Figure E.1). Each workflow similarly displays a splash screen, enabling users to configure the cross-linker selection, identification and quantitation settings, and the input file selection (Figure E.2). Each workflow enables batch processing of samples, using a table to list input files by sample (Figure E.3). A click event activates each cell in the table, creating a modal dialog to select the desired input file. Users may also toggle the cross-linker selection or the active profiles for spectral quantitation (Figure E.4). Furthermore, users may configure general, spectral, identification, or report settings, specializing various parameters for different experimental
methodologies (Figure E.5). After running a batch workflow, users may then export the cross-link document and an Excel summary (Figure ??). Finally, each document viewer provides clean visualizations of data at various levels from a sample, using a tree view linked to a multipurpose graph. This viewer is particularly useful to visualize the relative abundance of an isotopically-labeled population over a sample (Figure E.6) or series of samples (Figure E.7). By providing a clean, minimalist design to process and visualize XL-MS data, our application simplifies cross-link identification and quantitation, amenable for any user.

2.4 Discussion

Cross-linking mass spectrometry provides topological information about protein complexes and protein-protein interactions in vivo, complementing existing structural biology techniques. By designing a platform to automate the identification and quantitation of cross-linked peptides, we streamlined what previously required copious amounts of manual work. Although other programs have been developed for the identification of MS-cleavable cross-linked peptides since we started work on XLTools [45], these competing platforms work with only a single cross-linker. Furthermore, they implement their own search databases for peptide identification, which identifies less proteolytic peptides contributing to cross-links in our internal testing than Protein Prospector. Consequently, leveraging well-tested technologies enhances the number of cross-links that may be identified, and simplifies XLTools's internal design.
Chapter 3

Algorithm Development for
Cross-Linked Peptide Quantitation

3.1 Introduction to Quantitative Proteomics

In addition to identifying the proteins within a sample, quantitative MS also determines the abundance of each component, facilitating differential analysis between samples. The general workflow involves preparing a sample, labeling the proteins with a chemical or metabolic reagent, mixing samples for multiplexed analysis, and quantifying the relative abundance between target samples. The field has evolved where even label-free spectral counting was used to quantify the tissue-specific expression of proteins proteome-wide [67]. Quantitative mass spectrometry therefore robustly measures protein expression levels for labeled and unlabeled samples.
Figure 3.1: Generalized workflow for quantitative MS.
Labeled proteins are cross-linked and digested with proteolytic enzymes, submitted for multistage MS analysis, and relative abundance quantified by the intensity ratio between different populations.

Numerous algorithms exist to accurately detect peptide abundance. Semiquantitative metrics count the presence or absence of a reaction and integrate over all samples within the experiment [68, 69]. Counting may be done for selected reactions (spectral counting) or database identifications (peptide counting), providing a basis for differential analysis. Modern workflows typically use intensity-based algorithms, which quantify the intensity of the precursor ion from selected reaction monitoring (SRM) [70]. Accurate peak selection from extracted ion chromatograms (XICs) for monitored reactions is crucial for accurate quantitation, and depends on spectral features like peak shape and the co-elution of labeled and unlabeled populations [70]. By virtue of the similarities in cross-linked peptide and standard peptide identification, we can apply similar approaches used for quantitative proteomics to the analysis of cross-linked peptides.

Determining the relative propensity of cross-link formation between samples can refine structural models and identify protein conformational states. Protein cross-linking is dependent on the surface accessibility of the linked residues and the SAS distance between them.
Significant changes in cross-link abundance between populations therefore reflect local structural folding or unfolding, or greater topological changes within the protein complex.

3.2 MS1 Quantitation Workflow

After identifying the cross-links within a sample, we determine the elution window for each cross-link and integrate the intensity over the time. We use a configuration file to specify constant modifications for each isotopically-labeled population, which we use to determine the mass of each unlabeled and labeled cross-link. We convert our cross-link masses to a target m/z list for ion extraction, and reconstruct a mass chromatogram for each m/z value (Figure 3.2). We then identify the elution window of our cross-link and integrate under the curve to calculate peak area. However, the low abundance of cross-linked peptides leads to significant peak distortion and low signal-to-noise ratios (Figure 3.3). We subsequently describe various peak-picking algorithms we applied to accurately differentiate analyte from contaminant in cross-linked peptide mass chromatograms.

Figure 3.2: Conversion of raw spectra to XICs. Peaks from target m/z values are extracted over time to represent the intensity over time. In the case of multiple peaks within a desired mass tolerance, we sum the raw intensities and calculate the weighted average of the m/z values by the intensities.
3.3 Shape-Based Peak-Picking

Numerous existing mass spectrometry and signal processing algorithms use the correlation of the experimental chromatogram to a theoretical spectral line shape as a feature for accurate peak-picking [70–72]. Generally, this involves an iterative fit to a Gaussian or Lorentzian function guided by a non-linear least squares fit, which we implemented using SciPy’s \texttt{curve_fit} function [73]. Unfortunately, the low signal-to-noise ratio and significant distortions from ideal peak shape led to poor peak detection, frequently failing to detect peaks and at best over-estimating the cross-link elution window while underestimating its max area (Figure 3.4, F.1 - F.2). Furthermore, our algorithm was unable to find solutions for peaks with split elutions or high-intensity contaminants, demonstrating the sensitivity of shape-based algorithms to complex spectra (Figure F.3).
Figure 3.4: Iterative fit to a Gaussian function.
Fit of an experimental chromatogram using 800 iterations.

Using a weighted, non-linear least-squares regression score to optimize curve fitting can improve peak detection in high noise backgrounds [74]. A least-squares regression applies weights to each Euclidean distance between the experimental and fit curve, optimizing the fit function to minimize the variance for highly relevant sections of the curve. Owing to the high noise baseline present in mass spectrometry data, we assumed the relevance each point scaled as a polynomial function of the intensity, which should add resilience to intensity fluctuations of low magnitude. However, the resulting correlation between experimental mass chromatograms and the fit Gaussian functions remained poor, retaining the deficiencies of the unweighted model (Figure 3.5, F.1 - F.3). Due to the inherent noise in mass spectrometry data and the low abundance of cross-linked peptides, we determined that simple, shape-based peak-picking algorithms were not amenable to quantitative XL-MS analysis.
3.4 Wavelet-Based Peak-Picking

Continuous wavelet transformation (CWT) fits a continuous function into discrete wavelets, and the coefficients of each wavelet differentiate signal from noise for peak filtering and picking. CWT is routinely applied to intrinsically noisy data, such as seismological waves [75], and has been successfully applied for low-resolution MS peak smoothing and processing [76]. The Mexican Hat wavelet (the second derivative of the Gaussian function) provides a spectral-like line shape, but does not assume a constant baseline, making it ideally for MS data [71, 76, 77]. Identifying temporal regions of high-amplitude in the wavelet-transformed data in both coarse (low-frequency) and fine (high-frequency) wavelets detects peak elution time, providing more resilience to noise than simple shape-based curve fitting.
We applied SciPy’s `find_peaks_cwt` with the built-in ricker (Mexican Hat) wavelet to assess the accuracy of CWT for mass chromatogram peak detection. Although CWT performed well for simple peak-picking, increasing spectral complexity resulted in decreasing accuracy, particularly with high-noise baselines (Figure 3.7, F.1 - F.3). We therefore sought alternative algorithms or a multi-feature model to improve our peak-picking approach to increase the number of quantifiable cross-linked peptides per sample.
Figure 3.7: CWT fit to MS data. The relative enrichment of yellow in the region from 2.0-2.5 on the X-axis signifies it contains the cross-link elution, consistent with manually curated data.

3.5 Simple Peak-Picking

While seeking out non-canonical peak-picking algorithms for cross-link quantitation, we found a paper describing a simple, shape-independent algorithm applicable for labeled and unlabeled analyte quantitation in MS data [8]. The algorithm selects local maxima above a certain intensity threshold as candidate peaks. The peaks are then iteratively processed by descending intensity to identify the peak start and points for quantitation. The baseline of each peak is calculated as a percentage of peak intensity or dependent on the signal-to-noise ratio (SNR), and the first local minima outside the baseline defines the peak bounds (Figure 3.8). Due to the lack of assumptions about peak shape, we calculate the area using trapezoidal integration of intensity over the elution window.
Figure 3.8: AB3D algorithm.
The algorithm finds each local maximum, and the peak elution window is identified by finding the first local minima below a certain intensity threshold. Here, the intensity threshold is set to 50% (a) or 30% (b) of the local maximum. Adopted from Aoshima et al. (Public Domain) [8].

We applied a SNR-based AB3D algorithm to compare the accuracy of AB3D to CWT for cross-linked peptide peak detection in mass chromatograms. Although both algorithms correctly identify the start and end points of peaks in clean mass chromatograms with similar accuracy, AB3D substantially outperformed CWT for peak detection in chromatograms with noisy baselines or split elutions. However, although AB3D robustly detected peaks from the baseline, it cannot differentiate nearly isobaric contaminants from analytes (Figure 3.3), necessitating peak validation prior to quantitation.
3.6 Peak Validation

To validate candidate peaks, we employed a classification and regression tree (CART) model trained on more than 350 manually curated and tagged XICs from *in vitro* and *in vivo* cross-linked 26S proteasome data acquired on multiple high-resolution mass analyzers. Each mass chromatogram was tagged by the peak validity, mass error (PPM), isotope correlation (dotp), peak size, and mass correlation (mass $\rho$) over a manually selected elution window (Section F.4). We trained the data using SciKit Learn’s decision tree model, to generate a classifier that could accurately differentiate analyte from contaminant. Our model achieved over 95% accuracy on both our training and testing sets, with a false positive rate of 3.5% and a false negative rate of 2.8%, demonstrating the high selectivity of our peak validation approach even with high spectral complexity.

Figure 3.9: Tree classifier schematic.
Simplified schematic demonstrating the simplified classification of spectra based on spectral parameters.

We also optimized various parameters for AB3D and CWT peak-picking, and independently evaluated the model accuracy by assessing the similarity between the predicted and manually curated integrated peak areas (Figure 3.10). We added a penalty to our scoring function, subtracting the area integrated outside the bounds of our true XIC (Equation 3.1). Consequently, our model training, peak-picking refinement, and validation
therefore derive from optimizing the fraction of true area that is calculated to a reference set. Overall, our AB3D algorithm was more accurate at all spectral complexities, correctly predicting over 90% of the area for clean chromatograms and split elutions, forming the basis of our peak-picking approach.

\[
    \text{score} = \frac{\text{predicted} - \text{error}}{\text{actual}} \tag{3.1}
\]

Figure 3.10: Scoring visualization.
Visualization of how to calculate the predicted and error area for Equation 3.1.

3.7 Discussion

Quantitative cross-linking mass spectrometry enhances our understanding of the structural changes that occur between populations, complementing semi-quantitative approaches like peptide counting. However, due to the low abundance and high noise of cross-linked
peptide spectra, cross-link quantitation has required manual peak-picking, preventing high-throughput analysis. We developed peak-picking algorithms specialized to unlabeled, low SNR, high-resolution spectra that provide accuracy in excess of 95%, enabling the streamlined quantitation of cross-linked peptides in proteomics workflows.

Furthermore, our model removes many assumptions in existing standard quantitative proteomics approaches while retaining high fidelity, demonstrating the power of a simple model. Most proteomics-based peak-picking algorithms use peak-shape and isotope intensity correlation to a reference peptide as core parameters for their model [70, 78]. These assumptions only hold for high-quality data and rapidly deteriorate as spectral complexity increases. These assumptions had low predictive power in our training set: the correlation of our experimental spectra to an isotope envelope (mass $\rho$) was not included in our classification tree (Figure F.4), and peak fitting to a Gaussian or Lorentzian function poorly modeled real mass chromatograms (Figure 3.4 - 3.5). Previous studies have also suggested these theoretical profiles may be instrumentation-dependent, which would limit their generalizability [70, 79]. In contrast to standard peak-picking algorithms, our approach only assumes high mass accuracy to a target m/z value and a similar elution profile for all isotopes within the isotope envelope. Despite the high complexity of our experimental mass chromatograms, our trained model reduces the number of assumptions required for classification while retaining high accuracy, potentially expanding its applicability to a wide range of high-resolution MS instruments.
Chapter 4

XL-MS Analysis of the 26S Proteasome

4.1 Significance of the 26S Proteasome

Protein degradation is crucial for diverse biological processes, including cell-cycle regulation and oxidative response [80–82]. The ubiquitin-proteasome system (UPS) is the core targeted protein degradation machinery in cells, and degrades over 80% of all intracellular proteins [83]. The 26S proteasome, the degradation machinery of the UPS, consists of a catalytic core (the 20S) and lid (19S), which mediates protein entry into the core [84]. Dysregulation of the UPS has profound implications in human disease, being implicated in diverse conditions from neurodegenerative diseases to cancer [85, 86]. Various pharmacological agents, which globally inhibit proteasome activity, have been approved for the treatment of cancer [87]. Due to adverse side effects of globally inhibiting the UPS, the discovery of pharmaceuticals modulating a subset of UPS functionality should reduce drug side effects and improve patient prognosis.
We therefore sought to identify structural changes within the 26S proteasome upon oxidative stress. Prior work suggests that oxidative stress leads to disassembly of the 26S holo-complex, leading to increased 20S activity to degraded oxidized proteins [88]. Consequently, oxidative stress likely induces conformation rearrangements within the 26S culminating in proteasome disassembly. The heterogeneity, flexibility, and size of the oxidized 26S proteasome prevents its analysis by traditional structural methods like XRD or NMR. We therefore sought to apply quantitative XL-MS to identify structural changes between oxidized and native proteasome, using the MS1-level quantitation workflow described in chapters 2 and 3.

4.2 Experimental Design

26S proteasome was affinity purified from HEK-293 cell lines expressing tagged proteasome baits (Rpn11, Rpt6, and Pre10) as described previously [12]. Cells were grown in replicates, one sample natively and the other using stable isotope labeling with amino acids in cell culture (SILAC) for metabolic protein labeling. The samples were then cross-linked using disuccinimidyl sulfoxide (DSSO), a homobifunctional, MS-cleavable, lysine-reactive cross-linker, digested with trypsin, and the proteolytic peptides mixed prior to LC-MS analysis. Cells were cross-linked in vivo (in live cells), in vitro (on-bead during purification), and semi vivo (on-bead after formaldehyde-treatment of cells). We acquired two biological replicates per cell line, and performed two technical replicates per sample.

We acquired the data using a Thermo Scientific Easy-nLC 1000 UPLC system coupled to an Orbitrap XL mass analyzer as previously described [57]. We extracted the peak lists using a Raw_Extract script from XCalibur v2.4 (Thermo Scientific), and submitted the MS\(^3\) data for database searching by Protein Prospector using Batch-Tag against a concatenated decoy database [6, 57]. The resulting peak lists and peptide search summaries were then
submitted to XLTools for cross-linked peptide identification and quantitation. Candidate regions undergoing structural rearrangements were identified by the fold change of cross-links between the populations.

![Figure 4.1: Experimental design for 26S proteasome analysis. Cells were grown in replicate, with one population labeled and the other population unlabeled. The 26S was affinity purified and cross-linked prior to mixing, and submitted for LC-MS analysis. Candidate cross-links that were changed between populations are visualized using a volcano plot.](image)

### 4.3 Methods & Results

We identified over 1450 unique proteasome-containing cross-links over all baits and replicates, and were able to quantify over 850 of these linkages. Of these, only a small fraction (18%-36%) of linkages were similar between different affinity purification baits (Table G.1). However, the low identification reproducibility between baits was similarly observed between biological replicates, possibly suggesting only a fraction of cross-links in each sample were identified due to sample complexity (Table G.2). Despite the initial concerns of identification reproducibility, the quantitation results were highly reproducible between biological replicates (Figure G.1), demonstrating the advantage of quantitative versus semiquantitative approaches.
We subsequently identified significantly changed linkages using a 1-sample T-test, filtering for linkages below a p-value of 0.05 and above a 2-fold change in magnitude ($|\log_2(ratio)| > 0.58$). We performed our analysis using the software package PaDua on the difference between log2-transformed labeled and unlabeled cross-link L/H ratios [89]. We visualized the using a volcano plot to determine the general distribution of between populations. However, we noticed significant skew in our cross-link ratios (Figure 4.2a, G.2a, G.3a), which we speculated was due the relative change in 19S and 20S abundance between oxidized and non-oxidized samples after affinity purification. We therefore calculated the abundance of proteasome and proteasome-interacting proteins using MaxQuant and MS/MS quantitation on our SILAC samples for each biological replicate [90]. Normalizing cross-link abundance to protein abundance removed the skew from our data, most notably for 19S interactions from Pre10- (a 20S subunit) purified samples (Figure 4.2b, G.2b, G.3b). The significantly changed abundance of both standard and cross-linked peptides corroborates the role of oxidative stress in inducing conformational rearrangements within the 26S proteasome and the eventual disassembly of the holocomplex.

![Volcano plots](image)

(a) Pre10 Pre-Normalization  
(b) Pre10 Normalized

Figure 4.2: Volcano plot of Pre10 linkages.  
Volcano plots for the fold change in the L/H cross-link abundance.
Owing to the varying coverage of quantifiable peptides in certain subunits of the 26S proteasome, we coarsened to 26S proteasome structure to 4 substructures, the alpha subunits, beta subunits, base, and lid (Table 4.1). The protein abundance was calculated as the average of log2-transformed values for all proteins contributing to a substructure. Each technical replicate was normalized to L/H protein ratio using MS/MS quantitation data emanating from the same biological replicate. We assumed that the least abundant protein in the oxidized proteasome would be the limiting factor for cross-link formation, leading us to normalize our cross-link ratios to the cross-link contributing protein with the lowest L/H ratio. We used the individual protein abundance for proteasome-interacting proteins, since they are not stoichiometric components of the 26S proteasome.

<table>
<thead>
<tr>
<th>Alpha</th>
<th>Beta</th>
<th>Base</th>
<th>Lid</th>
</tr>
</thead>
<tbody>
<tr>
<td>α1</td>
<td>β1</td>
<td>Rpt1</td>
<td>Rpn1</td>
</tr>
<tr>
<td>α2</td>
<td>β2</td>
<td>Rpt2</td>
<td>Rpn10</td>
</tr>
<tr>
<td>α3</td>
<td>β3</td>
<td>Rpt3</td>
<td>Rpn11</td>
</tr>
<tr>
<td>α4</td>
<td>β4</td>
<td>Rpt4</td>
<td>Rpn12</td>
</tr>
<tr>
<td>α5</td>
<td>β5</td>
<td>Rpt5</td>
<td>Rpn2</td>
</tr>
<tr>
<td>α6</td>
<td>β6</td>
<td>Rpt6</td>
<td>Rpn3</td>
</tr>
<tr>
<td>α7</td>
<td>β7</td>
<td>Rpn5</td>
<td>Rpn6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Rpn7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Rpn8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Rpn9</td>
</tr>
</tbody>
</table>

Table 4.1: Proteasome substructures. Substructures used to normalize cross-link abundance by protein abundance.

Due to compound error from imperfect peak-picking, peak validation, and sample variability, we decided to discard cross-link ratios with low reproducibility between biological
replicates. We performed outlier detection on the difference on the log2-transformed L/H ratios using SciKit Learn’s `EllipticalEnvelope`, which assumes a Gaussian-distributed dataset. We assumed a contamination rate of 10%, to minimize the effect of technical variation or error on our dataset.

We then mapped our changed cross-link ratios to reference 26S proteasome structures to elucidate potential structural changes that occur upon proteasome oxidation. We used three existing cryoEM proteasome models: a 4.35Åstructure (5GJQ) of the 26S proteasome bound to Usp14 using an *E. coli* expression system, a 3.9Åand 6.8Åstructure of natively-expressed 26S proteasome (5L4G and 5LN3, respectively) [91]. We compared the similarity between the existing models using the root mean square deviation (RMSD) between atoms in of aligned structures, using in-house PyMol scripts for structural alignment and RMSD calculation (Figure G.4). We noted that the *E. coli* structure (5GJQ) most closely resembled the high-resolution native proteasome structure (5L4G). We then calculated the theoretical Euclidean distance between α carbons on cross-linked lysines for both significantly changed and unchanged linkages, to determine how existing models corresponded to spatial constraints derived from our cross-linking data (Figure 4.3). Despite the similarity between the 5GJQ and 5L4G structures, our cross-linking data most closely resembled the native 26S proteasome structures, partially corroborating the validity of our cross-linking dataset. The theoretical Euclidean cross-link distance constraints on the 5GJQ structure were primarily above 30Å, suggesting our cross-links model a different conformational state of the 26S proteasome. Interestingly, we observed no difference between the distance distribution for changed and unchanged linkages for any reference model, potentially suggesting that subtle structural changes mediate 26S proteasome dissociation.
4.4 Discussion

We demonstrate a proof of concept for the application of automated cross-linked peptide identification and quantitation to the analysis of proteins cross-linked \textit{in vivo} and after affinity purification. This establishes the applicability of quantitative mass spectrometry to complex samples, an advancement over prior work using standard proteins [78, 92]. We are currently working to apply our derived cross-linking spatial constraints to enhance existing models and identify conformational states of the 26S proteasome.
Bibliography


56


Appendix A

General Algorithms

Supplementary algorithms utilized during the design and implementation of XLTools. Algorithms presented use naive pseudocode as a high-level abstraction from individual language implementations.

A.1 Pseudocode Conventions

Due to the many dialects of pseudocode available, to avoid ambiguity, we specify the exact pseudocode conventions used within our algorithms. Our conventions are partially inspired by both C and Python.

A.1.1 Comments

Single-Line Comment
Format: // Example of a comment.
Result: Do not process any code from the first // until the end of the line.

C-Style Comments
Format: /* Example of a comment.
  * This can be multi-line.
  */
Result: Do not process code between the markers /* ... */.

A.1.2 Operators

Assignment
Format: \( a = b \)
Result: Assign the variable \( a \) to the \( b \), by-value for simple types or by-reference for objects.

Relative
Format: \( a =\!\!\!= b \) // Is Equal To
\( a \neq b \) // Is Not Equal To
\( a < b \) // Is Less Than
\( a \leq b \) // Is Less Than or Equal To
\( a > b \) // Is Greater Than
\( a \geq b \) // Is Greater Than or Equal To
Result: Check if variables satisfy the relational operator.
A.1.3 Arguments

Default Argument
Format: \texttt{FUNCTION}(a = 1)
Result: Define function with positional argument of \(a\) with a default value of 1, similar to C++ syntax.

A.1.4 Branches and Loops

Conditional
Format:

\begin{verbatim}
if \ a == 0  
  \ A
elseif \ a == 1
  \ B
else
  \ C
\end{verbatim}

Result: Execute branch if and only if (iff) condition evaluates true. \texttt{elseif} conditions are evaluated only if all preceding conditions are false. If none of the conditions evaluate to true, and an \texttt{else} clause exists, execute \texttt{else}.

For Loop
Format:

\begin{verbatim}
for \ i = n_i \textbf{ to } n_j \textbf{ by } n_s
  \ A
\end{verbatim}
Result: Execute statement A for each value of i from \([n_i, n_j]\), increments i by 
\(n_s\) each loop. The keyword by may be omitted, and \(n_s\) defaults to 1.

While Loop

Format:

\[
\text{while } i > 0 \\
\quad \text{A} \\
\text{Result: Execute statement A as long as } i > 0.
\]

A.1.5 Mathematical

Procedures

Format:

\[
\text{while } i > 0 \\
\quad \text{A} \\
\text{Result: Execute statement A as long as } i > 0.
\]

Notation

Format: \(\text{ERF}(i)\)

Result: Call the Gauss error function for i. All procedures in the C standard 
library, like \text{ERF}, \text{POW}, are implicitly defined.
A.1.6 Objects

Structures

Format:

```c
struct Data {
    int x;
};
```

Result: Define a composite data type with the field \( x \), which is an integral value. The field \( x \) can be accessed from \( Data \) instance \( d \) via \( d.x \).

Array fields are denoted by `type*`, sets by `type**`, and dicts by `type***`.

A.1.7 Collections

Array

Format: let \( A[n_i..n_j] \) be a new array

Result: Define a fixed-length collection of length \( n_j - n_i + 1 \). The array is a structure which supports indexing and has a field \( length \), containing the number of items in the array. Array literals may also be defined using the Python-like syntax \( A = [1, 2] \).

Sets

Format: let \( S \) be a new set
Result: Define a collection of unique values. The set is a structure which supports item lookup, and item addition and deletion. The structure has a field *length*, containing the number of items in the set. Set literals may also be defined using the Python-like syntax $S = \{1, 2\}$.

**Dicts**

Format: let $D$ be a new dict

Result: Define a key-value collection of unique keys. The dict is a structure which supports item lookup, indexing, and item addition and deletion. The structure has a field *length*, containing the number of items in the dict. Dict literals may also be defined using the Python-like syntax $D = \{1 : 3, 2 : 4\}$.

**Indexing**

Format: $A[i]$

Result: Access value in array or dict $A$ at index or key $i$, respectively. The value may be assigned, using $A[i] = b$.

**Item Lookup**

Format: $i \in A$

Result: Check if $i$ is in the set $A$, or in the keys of the dict $A$. 

65
A.2 Utility

COPY-RANGE(Src, j, k, Dst, p)

// Copy from Src in the range [j, k] to Dst starting from p.
// input:
//   Src       Source array to copy from
//   j         First index in Src range
//   k         Last index in Src range
//   Dst       Destination array to copy into
//   p         First index in Dst range
// warning: Requires that Dst.length ≥ p + k - j

1  n = k - j
2  for i = 1 to n
3     B[p + i - 1] = A[j + i - 1]

COPY-ARRAY(A)

// Copy entire array and return copied array.
// input:
//   A        Array to copy
// output: Shallow copy of array

1  let B[1..A.length] be a new array
2  COPY-RANGE(A, 1, A.length, B, 1)
3    return B

IDENTITY(a)

// Identity transformation of object.

1  return a
### A.3 Numeric

**Binomal-Summation** \( (n, i_0 = 1, i_n = n) \)

```plaintext
// Calculate \( \sum_{k=i_0}^{i_n} \binom{n}{k} \)
sum = 0
for k = i_0 to i_n
    sum = sum + \binom{n}{k}
return sum
```

**Sum** \((A, i, j, \text{pred} = \text{Identity})\)

```plaintext
// Sum values over range \([i, j)\).
sum = 0
for k = i to j - 1
    sum = sum + \text{pred}(A[k])
return sum
```

**Average** \((A, i, j, \text{pred} = \text{Identity})\)

```plaintext
// Average values in A over range \([i_v, j_v)\).
length = j - i - 1
return \text{Sum}(A.i, j, \text{pred})/length
```
Weighted-Average($V, i_v, j_v, pred_v = \text{Identity}, W, i_w, j_w, pred_w = \text{Identity}$)

// Average values in $V$ over range $[i_v, j_v)$ by weights $W$ over range $[i_w, j_w)$.

1  sum = 0
2  weight = 0
3  distance = $j_v - i_v - 1$
4  for $k = 1$ to $distance$
5      sum = sum + pred_v(V[i_v + k]) \times pred_w(W[i_w + k])
6      weight = weight + pred_w(W[i_w + k])
7  return sum/weight

A.4 Statistics

Norm-Pdf($x$)

// Calculate PDF for norm distribution.

1  return $\exp(-x^2/2)$

Norm-Cdf($x$)

// Calculate CDF for norm distribution.

1  return $1 + \frac{\text{erf}(x/\sqrt{2})}{2}$

Cauchy-Pdf($x$)

// Calculate PDF for Cauchy distribution.

1  return $\frac{1}{\text{CAUCHY}_\text{CDF}(1+x^2)}$

Cauchy-Cdf($x$)

// Calculate CDF for Cauchy distribution.

1  return $0.5 \times \text{atan}(x) \times \text{CAUCHY}_\text{CDF}$
GAUSSIAN-Pdf($x, \bar{x}, \sigma$)

// Calculate PDF for Gaussian distribution.
1 return Norm-Pdf($\frac{x-\bar{x}}{\sigma}$)

GAUSSIAN-Cdf($x, \bar{x}, \sigma$)

// Calculate CDF for Gaussian distribution.
1 return Norm-Cdf($\frac{x-\bar{x}}{\sigma}$)

LORENTZIAN-Pdf($x, \bar{x}, \Gamma$)

// Calculate PDF for Lorentzian distribution.
1 width = $\frac{\Gamma}{2}$
2 return Cauchy-Pdf($\frac{x-\bar{x}}{\text{width}}$)

LORENTZIAN-Cdf($x, \bar{x}, \Gamma$)

// Calculate CDF for Lorentzian distribution.
1 width = $\frac{\Gamma}{2}$
2 return Cauchy-Cdf($\frac{x-\bar{x}}{\text{width}}$)
A.5 Searching

Min-Element\( \langle A, j, k, \text{pred} = \text{Identity} \rangle \)

// Get index for smallest value in \( A \) in range \([j, k)\).

1 // Return last index if range is empty.
2 if \( j == k \)
3 \hspace{1em}\text{return}\ k
4
5 smallest = first
6 for \( i = j \) to \( k - 1 \)
7 \hspace{1em}\text{if pred}(A[i]) < \text{pred}(A[smallest])
8 \hspace{2em}smallest = i
9 \text{return}\ smallest

Max-Element\( \langle A, j, k, \text{pred} = \text{Identity} \rangle \)

// Get index for smallest value in \( A \) in range \([j, k)\).

1 // Return last index if range is empty.
2 if \( j == k \)
3 \hspace{1em}\text{return}\ k
4
5 largest = first
6 for \( i = j \) to \( k - 1 \)
7 \hspace{1em}\text{if pred}(A[i]) > \text{pred}(A[smallest])
8 \hspace{2em}largest = i
9 \text{return}\ largest
LOWER-BINARY-SEARCH\((A, j, k, v, pred = \text{Identity})\)

// Get index for lower bound in A for value v in range \([j, k]\).
1 \hspace{1em} distance = k - j
2 \hspace{1em} \textbf{while} distance > 0
3 \hspace{2em} step = distance/2
4 \hspace{2em} i = j + step
5 \hspace{2em} \textbf{if} pred(A[i]) < v
6 \hspace{3em} j = i + 1
7 \hspace{3em} distance = distance - step + 1
8 \hspace{2em} \textbf{else}
9 \hspace{3em} distance = step
10 \hspace{2em} distance = k - j
11
12 \hspace{1em} \textbf{return} j
Upper-Binary-Search($A, j, k, v, \text{pred} = \text{Identity}$)

// Get index for upper bound in $A$ for value $v$ in range $[j, k]$.

1. $distance = k - j$
2. while $distance > 0$
3.   $step = distance / 2$
4.   $i = j + step$
5.   if $\text{pred}(A[i]) \geq v$
6.     $j = i + 1$
7.     $distance = distance - step + 1$
8.   else
9.     $distance = step$
10. $distance = k - j$
11.
12. return $j$

A.6 Combinations and Permutations

Combinations-Impl($A, p, q, k, C, i, T, count$)

// Recursively sample $k$-length subsets of $A$ from $[p, q]$ and add to $C$, using $T$ as a holder.

1. if $count == 0$
2.   $C[i] = \text{Copy-Array}(T)$
3.   return $i + 1$
4. else
5.   for $x = p$ to $q$
7.     $i = \text{Combinations-Impl}(A, p + 1, q, k, C, i, T, count - 1)$
Combinations($A, k$)

   // Sample all combinations for $C(A.length, k)$
   // input:
   //   $A$  Input array to sample combinations from
   //   $k$  Combination length
   // output: Array of all $k$-length combinations from $A$.

1  $n = \binom{A.length}{k}$
2  let $C[1..n]$ be a new array
3  let $T[1..k]$ be a new array
4  Combinations-Impl($A, 1, A.length, k, C, 1, T, k$)
5  return $C$
Appendix B

Abstraction & Standard Library

Supplementary materials for the implementation of a core abstraction platform and standard library for XLTools.

B.1 Filtering Streams

Sample specialization of a filtering streambuf to transform the input to represent each byte twice, sequentially.

```cpp
#include <pycpp/stream/filter.h>
#include <sstream>

/**<*
 * Double each byte in the input buffer.
 */
void doublechars(const void*& src, size_t srclen,
                 void*& dst, size_t dstlen,
                 size_t char_size)
{
```
size_t bytes = std::min(srclen, dstlen / 2) * char_size;
const char* src_ = reinterpret_cast<const char*>(src);
char* dst_ = reinterpret_cast<char*>(dst);

// copy bytes
while (bytes--) {
    *dst_++ = *src_;
    *dst_++ = *src_++;
}

// reassign to buffer
src = (const void*) src_;  
dst = (void*) dst_;  

int main()
{
    std::istringstream sstream("This is a message");
    filter_istream stream(sstream, doublechars);
    // "TThhiiss iiss aa mmeessssaaggee".
    std::cout << stream.rdbuf() << std::endl;

    return 0;
}
Appendix C

XL-MS Data Pipeline

Supplementary materials for the creation of an XL-MS data pipeline.

C.1 Constants

Constant, global values for XL-MS routines.

// Mass spectrometry
double NEUTRON_MASS = 1.0033548378
double PROTON_MASS = 1.00727647
double WATER_MASS = 18.010564942

// Chemical
AMINOACID_MASS = {
    'A': 71.0371137957,
    'C': 103.009184496,
    'D': 115.026943056,
}
\[
\begin{align*}
'E': & \quad 129.04259312, \\
'F': & \quad 129.04259312, \\
'G': & \quad 57.0214637315, \\
'H': & \quad 137.05891187, \\
'I': & \quad 113.084063988, \\
'K': & \quad 128.094963026, \\
'L': & \quad 113.084063988, \\
'M': & \quad 131.040484624, \\
'N': & \quad 114.042927463, \\
'P': & \quad 97.0527638599, \\
'Q': & \quad 128.058577527, \\
'R': & \quad 156.101111036, \\
'S': & \quad 87.0320284257, \\
'T': & \quad 101.04767849, \\
'U': & \quad 150.953634796, \\
'V': & \quad 99.0684139241, \\
'W': & \quad 186.079312961, \\
'Y': & \quad 163.063328554,
\end{align*}
\]

// Statistics

double CAUCHY_CDF = \pi;

double NORM_PDF = \sqrt{2\pi};
C.2 Models

The following objects are core models to help organize data for chemical cross-link sampling and matching.

/**
 * Peptide termini positions.
 */

```cpp
enum class Terminus : uint8_t
{
    NONE       = 0,
    // RELATIVE
    NTERM      = 1,
    CTERM      = 2,
    // ABSOLUTE
    PROTEIN_N_TERM = 3,
    PROTEIN_C_TERM = 4,
    PEPTIDE_N_TERM = 5,
    PEPTIDE_C_TERM = 6,
};
```

/**
 * Post-translational modification types.
 */

```cpp
enum class ModificationType : uint8_t
{
    CONSTANT    = 1,
    VARIABLE    = 2,
    ISOTOPE_LABEL = 4,
```
NON_CROSSLINKING = CONSTANT | VARIABLE | ISOTOPE_LABEL,
CROSSLINKING    = 8,
}

/**
 * Cross-Link quality classifiers.
 */
enum class LinkName: uint8_t
{
    STANDARD       = 0,
    LOW_CONFIDENCE = 1,
    FINGERPRINT    = 2,
    INCOMPLETE     = 3,
};

/**
 * Cross-Link type classifiers.
 */
enum class LinkType: uint8_t
{
    INTERLINK      = 0,
    INTRALINK      = 1,
    DEADEND        = 2,
    MULTILINK      = 3,
    SINGLE         = 4,
};

/**
 * Spectral peak definition.

/*
 * Param:
 * mz Mass to charge ratio
 * intensity Maximum intensity
 * z Charge state of the ion
 */

struct Peak
{
    double mz = 0;
    double intensity = 0;
    uint8_t z = 0;
};

/**
 * Definition for peak collection.
 */

typedef Peak* PeakList;

/**
 * \brief Search query from a given peptide ion.
 *
 * Param:
 * num Spectrum number
 * ms_level Level of the MS spectrum
 * rt Time of spectrum acquisition
 * parent_rt Time of parent spectrum acquisition
 * parent_mz Mass to charge value of parent
 * parent_intensity Intensity of parent ion
 */
* parent\_z 
  Charge of parent ion
* file 
  File of acquisition
* filter 
  Scan filter for MS acquisition
* peaks 
  MS spectral data (m/z, intensity, z)
* parent 
  Parent scan numbers
* children 
  Child scan numbers

/∗

struct Spectrum
{
    int num;
    uint8\_t ms\_level;
    double rt;
    double parent\_rt;
    double parent\_mz;
    double parent\_intensity;
    int parent\_z;
    char* file;
    char* filter;
    PeakList peaks;
    int* parent;
    int* children;
};

/∗∗

* Chemical formula definition.
*
* Param:
* string Chemical formula string
/* mass Mass of the molecule */

struct Formula {
    char* string;
    double mass;
};

/**
 * Definition for formula collection.
 */
typedef Formula* FormulaList;

/**
 * Post-translational modification definition.
 *
 * Param:
 * name Modification name
 * formula Chemical formula of the modification
 * aminoacid 1-letter codes of modified residues
 * terminus Termini modification reacts at
 * uncleaved Modification blocks proteolytic cleavage
 * active Modification is active
 * active_aminoacids Active subset of aminoacid
 * type Modification type
 * neutral_loss Mass loss during sequencing
 */

struct Modification
{
    char* name;
    Formula formula;
    char* aminoacid;
    Terminus terminus = Terminus::NONE;
    bool uncleaved = false;
    bool active = false;
    char* active_aminoacids = "";
    ModificationType type = ModificationType::VARIABLE;
    double neutral_loss = 0.;
};

/**
 * Definition for modification collection.
 */
typedef Modification* ModificationList;

/**
 * Cross-linker reactive site definition.
 *
 * Param:
 *    aminoacid    1-letter codes of modified residues
 *    deadend     Chemical formula change from dead-end
 *    terminus    Termini modification reacts at
 */
struct ReactiveSite
{
    char* aminoacid;
}
Formula deadend;
Terminus terminus;
};

/**
 * Definition for reactive site collection.
 */
typedef ReactiveSite* ReactiveSiteList;

/**
 * Cross-Linker object definition.
 *
 * Param:
 *   name Cross-Linker name, a unique identifier
 *   formula Chemical formula for the intact cross-linker
 *   fragment_list Fragment modification identifiers
 *   active Cross-linker is selected
 *   charge_list Possible charge states of the cross-linker
 */
struct Crosslinker
{
    char* name;
    Formula formula;
    bool active;
    ModificationList fragment_list;
    ReactiveSiteList site_list;
    int* charge_list = {0};
};
C.2.1 Mass Spectrometry Calculations

The following algorithms enable high-accuracy calculations for mass-spectrometry data.

\[ \text{Mz}(\text{mass}, \text{initial}_z = 0, \text{final}_z = 0) \]

// Calculate the mass-to-charge ratio for an ion.

// input:

// mass Initial m/z of the ion.

// initial\_z Initial charge of the ion.

// final\_z Final charge of the ion.

// output: Mass-to-charge ratio at final\_z.

1 if initial\_z \neq 0
2 mass = mass * initial\_z
3 mass = mass - (PROTON\_MASS \times initial\_z)
4 if final\_z \neq 0
5 mass = mass + (PROTON\_MASS \times final\_z)
6 mass = mass/final\_z
7 return mass
PPM(\texttt{theor\_mz, theor\_z, exper\_mz, exper\_z})

// Calculate the relative mass error between ions.

// input:

// \texttt{theor\_mz} Theoretical mass of ion.
// \texttt{theor\_z} Theoretical charge of ion.
// \texttt{exper\_mz} Experimental mass of ion.
// \texttt{exper\_z} Experimental charge of ion.

// output: Relative mass error between ions.
1 \texttt{theoretical} = \texttt{MZ(theor\_mz, exper\_z, theor\_z)}
2 \texttt{return (exper\_mz – theoretical)/theoretical}

DA(\texttt{theor\_mz, theor\_z, exper\_mz, exper\_z})

// Calculate the absolute mass error between ions.

// input:

// \texttt{theor\_mz} Theoretical mass of ion.
// \texttt{theor\_z} Theoretical charge of ion.
// \texttt{exper\_mz} Experimental mass of ion.
// \texttt{exper\_z} Experimental charge of ion.

// output: Absolute mass error between ions.
1 \texttt{theoretical} = \texttt{MZ(theor\_mz, exper\_z, theor\_z)}
2 \texttt{return exper\_mz – theoretical}

C.3 Cross-Linked Peptide Sampling

The following algorithms exhaustively sample subsets of the set $K$, which contains MS3 peptide identifications from an MS2 scan.
C.3.1 Objects

/**
 * Post–translational modification definition.
 */

struct Ptm
{
    Modification modification;
    int position;
};

/**
 * Definition for a flat post–translational modification collection.
 */

typedef Ptm* FlatPtm;

/**
 * Definition for flat PTM collection.
 */
typedef FlatPtm* FlatPtmList;

/**
 * Complete view of peptide’s post–translational mods.
 */

struct PtmInfo
{
    FlatPtm certain;
    FlatPtmList uncertain;
    double neutral_loss;
};
/**
 * Peptide search hit definition.
 *
 * Param:
 *  id  Accession number or mnemonic for protein
 *  name  Protein name
 *  peptide  Peptide sequence
 *  ppm  Mass error in PPM
 *  score  Peptide ion score
 *  ev  Expectation value for peptide identification
 *  start  Protein start position for peptide
 *  rank  Rank of identification
 *  preferred  Preferred name identifier
 *  modification  Post-translational modifications
 */

struct SearchHit
{

  char* id;
  char* name;
  char* peptide;
  char* ppm = 0;
  double score = 0;
  double ev = 0;
  int start = 1;
  int rank = 1;
  char* preferred;
  PtmInfo modification;
}
/**
 * Collection of peptide search hits.
 */
typedef SearchHit* SearchHitList;

/**
 * \brief Search query from a given peptide ion.
 *
 * Param:
 *　　fraction  File name of identification
 *　　rt  Retention–time of the query
 *　　num  Spectrum number of ion
 *　　mz  Mass to charge value of ion
 *　　z  Charge of ion
 *　　search_hit_list  Collection of peptide search hits
 */
struct SpectrumQuery
{
    char* fraction;
    int num = 0;
    double mz = 0;
    int z = 1;
    SearchHitList search_hit_list;
};
C.3.2 Routines

**Peptide-Sampling**($K$)

// Exhaustively sample all peptide combinations.

// input:

//   $K$ Array of peptides identified from a single cross-link ion

// output: Array of all peptide combinations from $K$.

```plaintext
1 \( n = \text{Binomial-Summation}(K.\text{length}, 1) \)
2 let \( C[1..n] \) be a new array
3 \( x = 1 \)
4 for \( i = 1 \) to \( K.\text{length} \)
5 \( D = \text{Combinations}(K, i) \)
6 \( \text{Copy-Range}(D, 1, D.\text{length}, C, x) \)
7 \( x = x + D.\text{length} \)
8 return \( C \)
```

\[
f(k, r) = \begin{cases} 
\infty, & \text{if } r \leq 1 \\
\lceil \frac{k-1}{r-1} \rceil, & \text{otherwise}
\end{cases} \tag{C.1}
\]

Figure C.1: Calculate the minimum number of cross-linkers with \( r \) reactive sites required to bridge \( k \) peptides.
\[ f(C, r) = \left\lceil \sum_{i=1}^{C.length} \frac{\text{REM}(C_i)}{r} \right\rceil \]  

Figure C.2: Calculate the minimum number of cross-linkers with \( r \) reactive sites for peptide subset \( C \) given a procedure to identify number of cross-linker remnants per peptide identification, \( \text{REM}(x) \).

\[
f(C, r) = \begin{cases} 
0, & \text{if } r \leq 1 \\
1 + \sum_{i=1}^{C.length} \left( \frac{\text{REM}(C_i)}{r} \right) - C.length, & \text{otherwise}
\end{cases}
\]  

(C.3)

Figure C.3: Calculate the maximum number of cross-linkers with \( r \) reactive sites for peptide subset \( C \).

### C.4 Cross-Linked Peptide Validation

The following algorithms validate theoretical cross-linked peptides by comparing their properties to the experimental data.

#### C.4.1 Objects

```*/
* Chemical cross-linker reactive end object.
*
* Param:
*     peptide_list Peptide-modified cross-linker ends.
*/```
∗ * dead_list Deadend cross–linker ends.  
∗ * count Intact cross–linker count.  
*/

```c
struct Ends {
    int* peptide_list;
    int* dead_list;
    int count;
};
```

```c
/** brief Cross–Linked peptide definition. */

struct Crosslink {
    int num;
    double mz;
    double rt;
    int z;
    SearchHitList hits;
    Ends ends;
    Crosslinker crosslinker;
    LinkName linkname;
    LinkType linktype;
};
```
C.4.2 Routines

**Flatten-Modification** (*modification*)

// Flatten modification to a single combination.

// input:

// *modification* Experimental post-translational modifications.

1 \( n_c = \text{modification}.\text{certain}.\text{length} \)

2 \( n_u = 0 \)

3 if \( \text{modification}.\text{uncertain}.\text{length} > 0 \)

4 \( n_u = \text{modification}.\text{uncertain}[0].\text{length} \)

5 let \( F[1..n_c+n_u] \) be a new array

6 for \( i = 1 \) to \( n_c \)

7 \( F[i] = \text{modification}.\text{certain}[i] \)

8 for \( i = 1 \) to \( n_u \)

9 \( F[n_c+i-1] = \text{modification}.\text{uncertain}[0][i] \)

10

11 return \( F \)

**Peptide-Sequence-Mass** (*sequence*)

// Calculate mass of amino acids in sequence.

// input:

// *sequence* Sequence of 1-letter amino acid codes.

1 \( \text{mass} = \text{WATER}.\text{MASS} \)

2 for \( i = 1 \) to \( \text{sequence}.\text{length} \)

3 \( \text{mass} = \text{mass} + \text{AMINOACID}.\text{MASS}[\text{sequence}[i]] \)

4

5 return \( \text{mass} \)
CrossLink-Theoretical-Mass(crosslink)

// Calculate the theoretical, neutral mass of the cross-link.

// input:

// crosslink Theoretical cross-linked peptide.

1 mass = 0

2 // Sum the sequence mass and non-cross-linking modification mass over all search hits.
3 for i = 1 to crosslink.hits.length
4     hit = crosslink.hits[i]
5     flat = Flatten-Modification(hit.modification)
6     mass = mass + Peptide-Sequence-Mass(hit.peptide)
7     for i = 1 to flat.length
8         modification = flat[i].modification
9         if modification.type ≠ ModificationType :: CROSSLINKING
10            mass = mass + modification.formula.mass
11
12 // Calculate the total cross-linker mass.
13 // The mass is the sum of the
14 crosslinker = crosslink.crosslinker
15 crosslinker_mass = crosslinker.formula.mass
16 crosslinker_count = crosslinker.ends.count
17 mass = mass + crosslinker_mass * crosslinker_count
18 for i = 1 to crosslinker.site_list.length
19    mass = mass + crosslinker.ends[i].mass * crosslinker.ends.dead_list[i]
20
21 return mass
CROSSLINK-ION-VALIDATION\((theor_{mz}, theor_{z}, exper_{mz}, exper_{z}, threshold = 20, threshold\_units = \text{PPM})\)

// Validate a candidate cross-link.

// input:

// theor_{mz}  Theoretical mass of cross-linked peptide.
// theor_{z}   Theoretical charge of cross-linked peptide.
// exper_{mz}  Experimental mass of cross-linked peptide.
// exper_{z}   Experimental charge of cross-linked peptide.
// threshold   Threshold for mass error.
// threshold\_units Units for mass error threshold.

// output: Theoretical ion is within tolerance of experimental.

1 if theor\_z == exper\_z
2   if threshold\_units == \text{DA}
3     return \text{DA}(theor\_mz, theor\_z, exper\_mz, exper\_z) < threshold
4 else
5     return \text{PPM}(theor\_mz, theor\_z, exper\_mz, exper\_z) < threshold
6 return \text{FALSE}

C.5  Cross-Linked Peptide Scoring

The composite score and the distribution of MS metrics used to weight cross-linked peptide identifications.

C.5.1  Metrics Distribution

The distributions of metrics used to generate the composite score. The raw metrics ("Data") from cross-link and peptide identifications from Rpt6-purified 26S proteasome sam-
amples was plotted against the best function fit to the distribution ("Fit"). We transformed the data using the inverse of common functions and spectral lineshapes (polynomial, exponential, Gaussian, Lorentzian), calculated a linear regression for the data, and selected the function with the best coefficient of determination ($R^2$).

Figure C.4: Expectation Value Distribution. Distribution of expectation values from more than 90,000 peptide identifications, generated from *in vitro* and *in vivo* cross-linked 26S proteasome samples, which is best modeled by an exponential distribution.
Figure C.5: Ion Score Distribution.
Distribution of ion scores from more than 90,000 peptide identifications, generated from \textit{in vitro} and \textit{in vivo} cross-linked 26S proteasome samples, which is best modeled by a Gaussian distribution.

Figure C.6: Minimum Ion Score Distribution.
Distribution of minimum ion score from more than 2,500 cross-link identifications, generated from \textit{in vitro} and \textit{in vivo} cross-linked 26S proteasome samples, which is best modeled by a Gaussian distribution.
Figure C.7: Maximum Ion Score Distribution.
Distribution of maximum ion score from more than 2,500 cross-link identifications, generated from \textit{in vitro} and \textit{in vivo} cross-linked 26S proteasome samples, which is best modeled by a Gaussian distribution.

Figure C.8: Cross-Link Ion PPM Distribution.
Distribution of relative mass errors (in PPM) from more than 7,000 cross-link identifications, generated from \textit{in vitro} and \textit{in vivo} cross-linked 26S proteasome samples, which is best modeled by a Lorentzian distribution.
Figure C.9: Peptide Ion PPM Distribution. Distribution of relative mass errors (in PPM) from more than 90,000 peptide identifications, generated from *in vitro* and *in vivo* cross-linked 26S proteasome samples, which is best modeled by a Lorentzian distribution.

C.5.2 Composite Score Equation

The composite scoring function resulting from combining multiple, independent identification metrics during cross-linked peptide identification.
Ion-Score-Value(crosslink)

// Calculate the ion score of the composite score.
// input:
// crosslink Cross-Link identification.
1 // Constants for the mean and stdev of the score.
2 mean = 23
3 stdev = 8.5
4 min_weight = 5
5 other_weight = min_weight/3
6
7 // Calculate the ion score value.
8 value = 0
9 min_index = MIN(crosslink.hits, key = score)
10 score = crosslink.hits[min_index].score
11 value = value + (min_weight * \text{GAUSSIAN-PDF}(score, mean, stdev))
12 for i = 1 to crosslink.hits.length
13 if i \neq min_index
14 value = value + (other_weight * \text{GAUSSIAN-PDF}(score, mean, stdev))
15 return value
PPM-Value(crosslink)

    // Calculate the PPM component of the composite score.
    // input:
    //     crosslink      Cross-Link identification.
    // Constants for the weight and sigma of the PPM.
1     peptide_weight = 1
2     crosslink_weight = 1
3     peptide_sigma = 20
4     crosslink_sigma = 30
5
6
7     // Calculate the PPM value.
8     value = 0
9     theor_mz = CrossLink-Theoretical-Mass(crosslink)
10    ppm = PPM(theor_mz, 0, crosslink.mz, crosslink.z)
11    value = value + weight * LORENZTIAN-Pdf(ppm, 0, crosslink_sigma)
12    for i = 1 to crosslink.hits.length
13       ppm = crosslink.hits[i].ppm
14    value = value + weight * LORENZTIAN-Pdf(ppm, 0, peptide_sigma)
15    return value
EV-Value(crosslink)

    // Calculate the EV component of the composite score.
    // input:
    //    crosslink  Cross-Link identification.

1    // Constants for the threshold and weight of the EV.
2    weight = -2.5/log10(1e-5)
3
4    // Calculate the EV value.
5    value = 0
6    for i = 1 to crosslink.hits.length
7        ev = crosslink.hits[i].ev
8        if ev < 0 ∨ ev > 1
9            ev = 1
10       value = value + weight * \sqrt{-log_{10}ev}
11    return value

Composite-Score(crosslink)

    // Calculate the overall quality score of the peptide identification.
    // input:
    //    crosslink  Cross-Link identification.

1    return Ion-Score-Value(crosslink) + Ppm-Value(crosslink) + Ev-Value(crosslink)

C.6 Spectral Quantitation

    Algorithms to extract ion intensities, construct XICs, and select the correct elution window for peak-picking.
C.6.1 Target Ion Extraction

Get-Peak-Mz(peak)

// Get mz field from peak.
1 return peak.mz

Get-Peak-Intensity(peak)

// Get intensity field from peak.
1 return peak.intensity

Sum-Peaks(A, i, j)

// Create peak from summed intensities of all peaks in range [i, j).
1 \( mz = \text{Weighted-Average}(A, i, j, \text{Get-Peak-Mz}, A, i, j, \text{Get-Peak-Intensity}) \)
2 \( intensity = \text{Sum}(A, i, j, \text{Get-Peak-Intensity}) \)
3 return Peak\{mz, intensity, 0\}

Max-Peaks(A, i, j)

// Get peak with highest intensity in range [i, j).
1 index = Max-Element(A, i, j, Get-Peak-Intensity)
2 return A[index]

Average-Peaks(A, i, j)

// Create peak from averaged intensities of all peaks in range [i, j).
1 \( mz = \text{Average}(A, i, j, \text{Get-Peak-Mz}) \)
2 \( intensity = \text{Average}(A, i, j, \text{Get-Peak-Intensity}) \)
3 return Peak\{mz, intensity, 0\}
**Extract-Target-Mz**\(\text{(peaklist, target\textunderscore mz, pred = Sum\textunderscore Peaks)}\)

// Extract all peaks from the list within a desired mass threshold of each target\textunderscore mz.

// input:

// peaklist Sorted list of peak objects, by m/z.
// target\textunderscore mz Sorted list of target m/z values.
// pred Predicate for how to process multiple matched peaks.

1 let \(T[1..\text{target\textunderscore mz.length}]\) be a new array

2 \(\text{first} = 1\)

3 for \(i = 1\) to \(\text{target\textunderscore mz.length}\)

4 \(mz = \text{target\textunderscore mz}[i]\)

5 \(\text{length} = \text{target\textunderscore mz.length}\)

6 \(\text{first} = \text{LOWER\textunderscore BINARY\textunderscore SEARCH}(\text{target\textunderscore mz}, \text{first}, \text{length}, mz, \text{Get\textunderscore Peak\textunderscore Mz})\)

7 \(\text{last} = \text{UPPER\textunderscore BINARY\textunderscore SEARCH}(\text{target\textunderscore mz}, \text{first}, \text{length}, mz, \text{Get\textunderscore Peak\textunderscore Mz})\)

8 if \(\text{first} == \text{last}\)

9 \(T[i] = \text{Peak}\{0, 0, 0\}\)

10 else

11 \(T[i] = \text{pred}(\text{target\textunderscore mz, first, last})\)

12 return \(T\)

---

**C.6.2 SPS Ion Extraction**

**Get\textunderscore Spectrum\textunderscore Num**\(\text{(spectrum)}\)

// Get num field from spectrum.

1 return spectrum.num
Iterate-SPS-Scans(crosslink, spectral_list, function, rt_delta = 90, z_delta = 1, isotope_delta = 2, thresh = 20, units = ppm)

// Iteratively process SPS scans, calling function on each suitable scan.

// Find upper and lower bound of the retention-time window.
length = spectral_list.length
rt = crosslink.rt
lower_rt = rt - rt_delta
upper_rt = rt + rt_delta
lower = LOWER-BINARY-SEARCH(spectral_list, 0, length, lower_rt, Get-Spectrum-Num)
upper = UPPER-BINARY-SEARCH(spectral_list, 0, length, upper_rt, Get-Spectrum-Num)

// Iterate over scan list and process all matching SPS scans.
for index = lower to upper
  spectrum = spectral_list[index]
  min_z = crosslink.z - z_delta
  max_z = crosslink.z + z_delta
  if min_z ≤ spectrum.z ≤ max_z
    sps_mass = MZ(spectrum.parent_mz, 0, spectrum.parent_z)
    crosslink_mass = MZ(crosslink.mz, 0, crosslink.z)
    for i = -isotope_delta to isotope_delta
      mass = sps_mass + NEUTRON_MASS * i
      if CrossLink-Ion-Validation(mass, 0, crosslink_mass, 0, thresh, units)
        function(spectrum)
Appendix D

Data Export

Supplementary materials detailing the various reports available within our OOXML export.

D.1 Identification Report

<table>
<thead>
<tr>
<th>Protein Accession</th>
<th>File</th>
<th>peptide_ID</th>
<th>peptide_length</th>
<th>peptide_scoring</th>
<th>score</th>
<th>peptide_sequence</th>
<th>cross-linked_peptide</th>
<th>cross-linked_peptide_length</th>
<th>cross-linked_peptide_scoring</th>
<th>cross-linked_peptide_score</th>
</tr>
</thead>
<tbody>
<tr>
<td>KRDH051009713</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KRDH051009714</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KRDH051009715</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KRDH051009716</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure D.1: Cross-Link identification report.

Each line consists of a cross-linked peptide identification, listing the file it was identified from, the peptides contributing to the identification, and various peptide and cross-link properties and scoring metrics.
D.2 Table Report

![Table Report](image_url)

**Figure D.2:** Cross-link table report.

The counts for each identification are displayed by file and linkage, facilitating comparison between individual files.
D.3 Quantitation Report

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scan m/z</td>
<td>Intensity 126</td>
<td>Intensity 127</td>
<td>Precursor m/z</td>
<td>Precursor z</td>
<td>Ratio</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>409</td>
<td>126.128.127.151</td>
<td>87.72</td>
<td>597.83</td>
<td>1.60</td>
<td>3.92</td>
<td>1.20</td>
</tr>
<tr>
<td>5</td>
<td>410</td>
<td>126.128.127.152</td>
<td>87.72</td>
<td>597.83</td>
<td>1.60</td>
<td>3.92</td>
<td>1.20</td>
</tr>
<tr>
<td>6</td>
<td>411</td>
<td>126.128.127.153</td>
<td>87.72</td>
<td>597.83</td>
<td>1.60</td>
<td>3.92</td>
<td>1.20</td>
</tr>
<tr>
<td>7</td>
<td>412</td>
<td>126.128.127.154</td>
<td>87.72</td>
<td>597.83</td>
<td>1.60</td>
<td>3.92</td>
<td>1.20</td>
</tr>
<tr>
<td>8</td>
<td>413</td>
<td>126.128.127.155</td>
<td>87.72</td>
<td>597.83</td>
<td>1.60</td>
<td>3.92</td>
<td>1.20</td>
</tr>
<tr>
<td>9</td>
<td>414</td>
<td>126.128.127.156</td>
<td>87.72</td>
<td>597.83</td>
<td>1.60</td>
<td>3.92</td>
<td>1.20</td>
</tr>
</tbody>
</table>

Figure D.3: SPS quantitation report.

Lists the raw abundances of each mass tag ion, or reporter ion, for all processed scans.
Appendix E

Qt5 Application

Supplementary figures displaying the Qt5 application for pre-defined workflows from the XLTools library.
E.1 Splash Menu

Figure E.1: Splash menu for XLTools.
Users can select from quantitative (Transitions Discoverer) and non-quantitative (CrossLink Discoverer) workflows to batch process XL-MS data, or select spectral (Spectral Viewer) or cross-link (CrossLink Viewer) document viewers.
E.2 Workflow Menu

Figure E.2: Splash menu for a workflow. Users can toggle input file selection, cross-linkers and isotope labels, add custom proteins for analysis, and configure general XL-MS settings.
E.3 Input File Table

Figure E.3: Interface to add files for batch analysis. Each row consists of a single MS experiment, containing the deisotoped spectra and peptide search summary.
E.4 Modification Selection

![Image of modification selection interface]

Figure E.4: Toggle post-translational modifications. Interface to toggle the active cross-linker selection and the isotopic and isobaric mass labels for spectral quantitation.
E.5 Settings Dialog

Figure E.5: Interface to add files for batch analysis. Each row consists of a single MS experiment, containing the deisotoped spectra and peptide search summary.
E.6 Cross-Link Viewer

Figure E.6: Mass tag abundance. Displays the value and variation of the abundance for an experiment utilizing 6 isobaric mass-tags.

Figure E.7: Mass tag abundance by experiment. Displays the abundance of each mass tag color-coded by experiment, for experiments utilizing 5 isobaric mass-tags.
Appendix F

Peak-Picking

Supplementary figures displaying the typical success of various peak-picking algorithms for spectra of differing quality.
F.1 Clean Chromatograms

Figure F.1: Algorithm accuracy for clean chromatograms. Displays the raw mass chromatogram (a), the unweighted, iterative fit (b), the weighted, iterative fit (c), and the peak detection using CWT (d). All algorithms perform well, although CWT performs the worst by breaking the singular peak into many wavelets at high-frequency.
F.2 Noisy Baseline

Figure F.2: Algorithm accuracy for noisy chromatograms. Displays the raw mass chromatogram (a), the unweighted, iterative fit (b), the weighted, iterative fit (c), and the peak detection using CWT (d). Only CWT detects the proper elution window, while the Gaussian fit algorithms almost exclusively model spectral noise.
F.3 Split Elution

Figure F.3: Algorithm accuracy for chromatograms with split peaks. Displays the raw mass chromatogram (a) and peak detection using CWT (b) for a cross-link that eluted over two windows. Only CWT determines the correct elution window. The Gaussian fit functions were unable to find a solution, demonstrating their sensitivity to complex chromatograms.

F.4 Peak Validation

Supplemental material for model generation to validate candidate peaklists. The various parameters were calculated as follows. We calculated mass $\rho$ using a Pearson’s correlation between the theoretical isotope abundance for an averaged aminoacid and the relative isotope abundance averaged over the elution window. We calculated the averaged dot product (dotp) based on intensity for all isotopes relative to the monoisotopic species. We estimated relative mass error by calculating the average of error between the theoretical and experimental m/z used for ion chromatogram extraction weighted by peak intensity. We estimated peak size by the number of MS1 points that were contained within the elution window, a rough estimator for the retention time difference.
Figure F.4: SciKit-Learn tree model. Optimal solution for our decision tree classifier. Attempts to use more max leafs overfit the training dataset, lowering the confidence of the validation set, through the repeated use of small variations of the $\text{dotp}$ regression to classify the data.

Figure F.5: Error rate. False positive and negative rate for our tree classifier model, demonstrating the fidelity of our model.
Appendix G

26S Proteasome Analysis

Supplementary materials for the quantitative cross-linking analysis of the 26S proteasome.

G.1 Replicate Reproducibility

<table>
<thead>
<tr>
<th></th>
<th>Rpn11</th>
<th>Rpt6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre10</td>
<td>0.18</td>
<td>0.27</td>
</tr>
<tr>
<td>Rpn11</td>
<td>-</td>
<td>0.36</td>
</tr>
</tbody>
</table>

Table G.1: Reproducibility between baits. Reproducibility of cross-link identifications between different baits. The reproducibility is reported as a percentage of the cross-links identified in both baits over the total number of cross-links identified in either bait.
<table>
<thead>
<tr>
<th>Bait</th>
<th>Sample 1</th>
<th>Sample 2</th>
<th>Reproducibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre10</td>
<td>SemiVivo 1</td>
<td>SemiVivo 2</td>
<td>0.41</td>
</tr>
<tr>
<td>Rpn11</td>
<td>SemiVivo 1</td>
<td>SemiVivo 2</td>
<td>0.24</td>
</tr>
<tr>
<td>Pre10</td>
<td>SemiVivo 1</td>
<td>SemiVivo 3</td>
<td>0.18</td>
</tr>
<tr>
<td>Pre10</td>
<td>SemiVivo 2</td>
<td>SemiVivo 3</td>
<td>0.38</td>
</tr>
<tr>
<td>Rpt6</td>
<td>SemiVivo 1</td>
<td>SemiVivo 2</td>
<td>0.36</td>
</tr>
</tbody>
</table>

Table G.2: Reproducibility of cross-link identifications between biological replicates.
Figure G.1: The relative quantitative reproducibility between biological replicates. The reproducibility is measured as a linear regression for the light-to-heavy (L/H) ratios of a cross-link between replicates.
G.2 Change in Cross-Link Abundance

Figure G.2: Volcano plot of Rpn11 linkages.
Volcano plots for the fold change in the L/H cross-link abundance.

Figure G.3: Volcano plot of Rpt6 linkages.
Volcano plots for the fold change in the L/H cross-link abundance.
G.3 Cross-Link Mapping to Reference Structures

Figure G.4: Correlation between proteasome structures. The RMSD between various models of the 26S proteasome. Two models of 5LN3 were used, one denoting a structure provided during initial model generation (initial) and one corresponding to the published structure (final).