GlycoProteomics Analysis Toolbox

GlycoPAT

Manual

Department of Chemical and Biological Engineering,
Clinical Translational & Research Center
State University of New York
University at Buffalo, Buffalo, NY 14260, USA

December 2016

Contents

1. Introduction

This document describes a MATLAB based toolbox and graphical user interface (GUI) called **GlycoProteomics Analysis Toolbox** (**GlycoPAT**). It aims to provide a platform for the streamlined analysis of traditional LC-MS^n based high-throughput experimental data for the identification of site-specific N- and O-linked glycosylation on various proteins. It follows a three-step workflow that includes:

- The *in silico* generation of a library of putative glycopeptides by protease digestion;
- Tandem MS analysis using a novel MS scoring scheme; and
- A spectrum and fragmentation viewer to visualize and annotate the results.

The unique features of the suite of programs include:

- The "SmallGlyPep" (SGP1.0) nomenclature: This new nomenclature defines a minimal representation of glycopeptides in linear text format. It contains all the necessary information required for the above MS data analysis.
- Ensemble score (ES-score): This single parameter scores the glycopeptide hits based on cross-correlation analysis, Poisson probability and decoy database based calculated *P*-values and additional considerations. Additionally, an acceptable ES-score for any experiment is calculated based on false discovery rate (FDR) calculations.
- User-extensible modular environment: The program provides a comprehensive set of functions for mass spectrometry data handling, computing glycopeptide mass/formula/fragmentation, glycopeptide digestion, spectra scoring, and MS/MS spectrum annotation. These functions are modular and thus they can be used in additional programs independent of GlycoPAT.

2. Download

GlycoPAT is a free open-source MATLAB toolbox. It is supported in Windows, Linux and Mac OS platforms. It is available from http://VirtualGlycome.org/GlycoPAT.

3. Installation

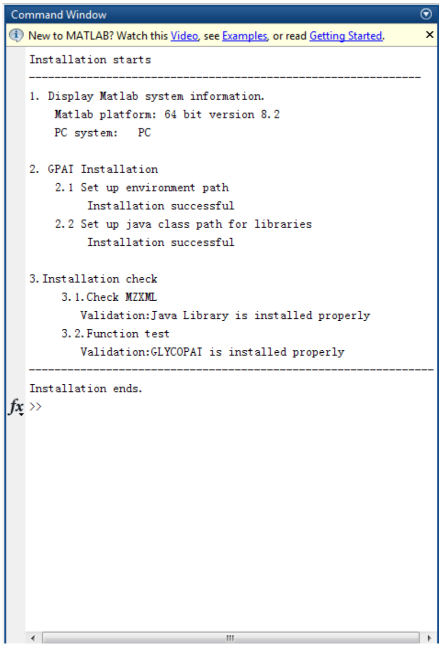
3.1 Stand alone application installation

To install the pre-compiled version of GlycoPAT, download the software, un-compress if necessary and double-click to initiate. The GUI version of GlycoPAT will open up. All GUI functions described below will be operational. To enable command line features follow instructions in the next section.

3.2 Installation of GlycoPAT in MATLAB environment

This section provides step-by-step instructions for the installation of GlycoPAT in 64-bit Windows OS. Similar steps can be followed for Linux and Mac OS platforms also. The program installed in this manner can be used to startup the GlycoPAT GUIs. It also allows the user to write short scripts to customize GlycoPAT functions.

1. **Software download:** Download "glycopat.zip" from site specified under 'Download' and unzip files in desired application directory (say C:\glycopat). One folder ("toolbox") and two files including the "GlycoPAT_Manual.pdf" and "installglycopat.m" file will appear. "toolbox" contains source code, example files and java libraries.
2. **MATLAB Startup**: Start MATLAB as administrator. MATLAB version 8.2 (R2013b) or higher is recommended. To open MATLAB as administrator, right click MATLAB icon on desktop or start menu, and choose "run as administrator". In the MATLAB command window, change current path to the GlycoPAT directory (say: C:\glycopat).
3. **Run Installation Command**: Type "`installglycopat`" and press enter to start installation. This: **i)** clears all variable from the MATLAB workspace, and displays MATLAB version and system information; **ii**) adds "toolbox" folder and its sub-directories to MATLAB search path; **iii**) adds GlycoPAT java library path to MATLAB dynamic java class path; **iv**) checks if GlycoPAT is installed properly. The procedure ends with an installation status report (see below).


Command Window
New to MATLAB? Watch this Video, see Examples, or read Getting Started.

```
Installation starts
---------------------------------------------------------------
1. Display Matlab system information.
    Matlab platform: 64 bit version 8.2
    PC system:   PC

2. GPAI Installation
    2.1 Set up environment path
         Installation successful
    2.2 Set up java class path for libraries
         Installation successful

3.Installation check
     3.1.Check MZXML
        Validation:Java Library is installed properly
     3.2.Function test
        Validation:GLYCOPAI is installed properly
-----------------------------------------------------------------
Installation ends.
>>
```

MATLAB window when GlycoPAT Toolbox is successfully installed.

Now the user can start to use all GlycoPAT commands, write user-customized functions, or open the primary GlycoPAT GUI by typing ‘glycopatgui’ at the command prompt.

4. **Change of MATLAB java class path**: While GlycoPAT is fully functional at the time of initial installation, loading of Java library in subsequent MATLAB startups is not automatic unless one of two alternate steps outlined below is carried out:

Option1 : Change of MATLAB java static path:

To do this, the Java library path names need to be appended to the “classpath.txt” file. In order to perform this operation:

- Type “`edit classpath.txt`” in MATLAB command window.
- Append the following java library name to this file. Here <GlycoPATInstallationDirectory> stands for the absolute directory path (e.g. C:\glycopat).

```
<GlycoPATInstallationDirectory>\toolbox\javalibs\jrap-ext-1.2.j
ar
```

Option 2: Change of MATLAB java dynamic path:

Change the ‘startup.m’ file so that the dynamic Java class path is loaded at the time of MATLAB start up. This involves:

- Opening the ‘startup.m’ file located in the GlycoPAT “\toolbox\startup” folder. Edit <GlycoPATInstallationDIR> in the tenth line of this file: {`GlycoPATpath='<GlycoPATInstallationDIR>'`} to reflect the absolute directory path (e.g. if the absolute directory path is ‘C:\glycopat’; replace this line with `GlycoPATpath='C:\glycopat'`).
- Locate the MATLAB startup folder by typing “userpath” in the MATLAB command window, or alternatively use “userpath(newdirectory)” to setup a new MATLAB startup folder location. Copy ‘startup.m’ into the MATLAB startup folder. If a “startup.m” file already exists in this location, append the contents of the GlycoPAT “startup.m” file into the existing “startup.m” file.

<u>**Note**</u>: Please see the FAQ section for further questions regarding software installation.

3.2 FAQ

1. Is GlycoPAT compatible with 32-bit systems?

Our programs have been developed with the viewpoint that most of the newer computers are 64-bit systems. Thus, the 64-bit Java library has been provided in GlycoPAT.

Limited testing has been performed on 32-bit computers and issues were not noted. However, in this case, it is better to use the 32-bit Java compiler for older computers.

2. What versions of MATLAB support GlycoPAT?

MATLAB version 8.2 (R2013b) or higher is recommended.

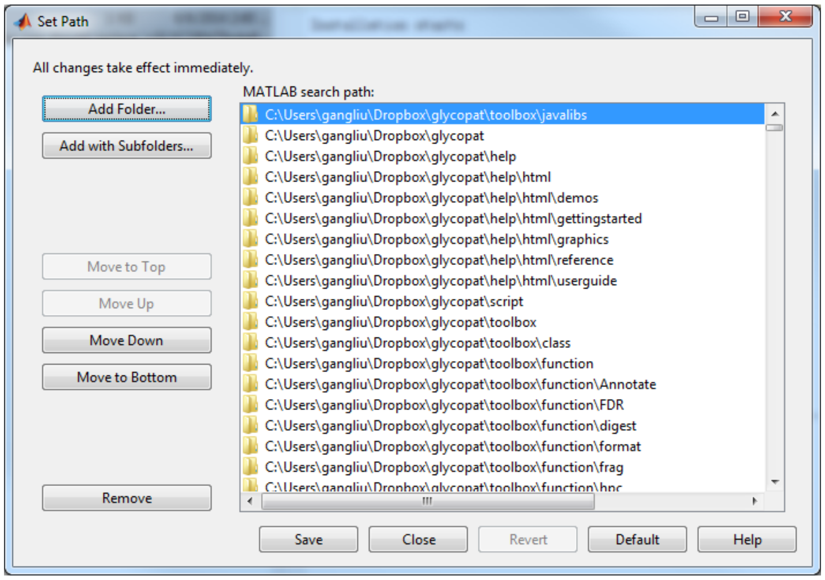
3. What will happen if I am not computer administrator when GlycoPAT is installed?

GlycoPAT will install properly at the time of first installation. However, problems may be encountered upon restarting MATLAB since MATLAB search path has not been saved. Errors may thus appear stating that "file cannot be found" or "function cannot be found".

Two alternative methods can be used to solve the issue:

A. Go to GlycoPAT installation directory and simply rerun the "installglycopat" command again.

B. Choose "Set Path" window from the 'File' pull-down menu in MATLAB (see below). Select the "Add with Subfolders…" option and browse to select the <User-defined directory> directory (e.g. C:\glycopat). Save the new path/settings, if the computer allows this.



All changes take effect immediately.

MATLAB search path:

```
C:\Users\gangliu\Dropbox\glycopat\toolbox\javalibs
C:\Users\gangliu\Dropbox\glycopat
C:\Users\gangliu\Dropbox\glycopat\help
C:\Users\gangliu\Dropbox\glycopat\help\html
C:\Users\gangliu\Dropbox\glycopat\help\html\demos
C:\Users\gangliu\Dropbox\glycopat\help\html\gettingstarted
C:\Users\gangliu\Dropbox\glycopat\help\html\graphics
C:\Users\gangliu\Dropbox\glycopat\help\html\reference
C:\Users\gangliu\Dropbox\glycopat\help\html\userguide
C:\Users\gangliu\Dropbox\glycopat\script
C:\Users\gangliu\Dropbox\glycopat\toolbox
C:\Users\gangliu\Dropbox\glycopat\toolbox\class
C:\Users\gangliu\Dropbox\glycopat\toolbox\function
C:\Users\gangliu\Dropbox\glycopat\toolbox\function\Annotate
C:\Users\gangliu\Dropbox\glycopat\toolbox\function\FDR
C:\Users\gangliu\Dropbox\glycopat\toolbox\function\digest
C:\Users\gangliu\Dropbox\glycopat\toolbox\function\format
C:\Users\gangliu\Dropbox\glycopat\toolbox\function\frag
C:\Users\gangliu\Dropbox\glycopat\toolbox\function\hpc
```

4. I have more questions. Where can I get help?

Write to the developer: neel@buffalo.edu

4. Getting help on GlycoPAT

Help documentation for GlycoPAT can be obtained using the following:

1. This “GlycoPATManual.pdf” manual has examples of GUI and command line functions.

2. A list of available functions is provided in Chapter 7. Type “help <functionName>” in MATLAB command window to obtain full description, usage and associated syntax.

- **Example 4.1: To get help on the digestion function of GlycoPAT use:**

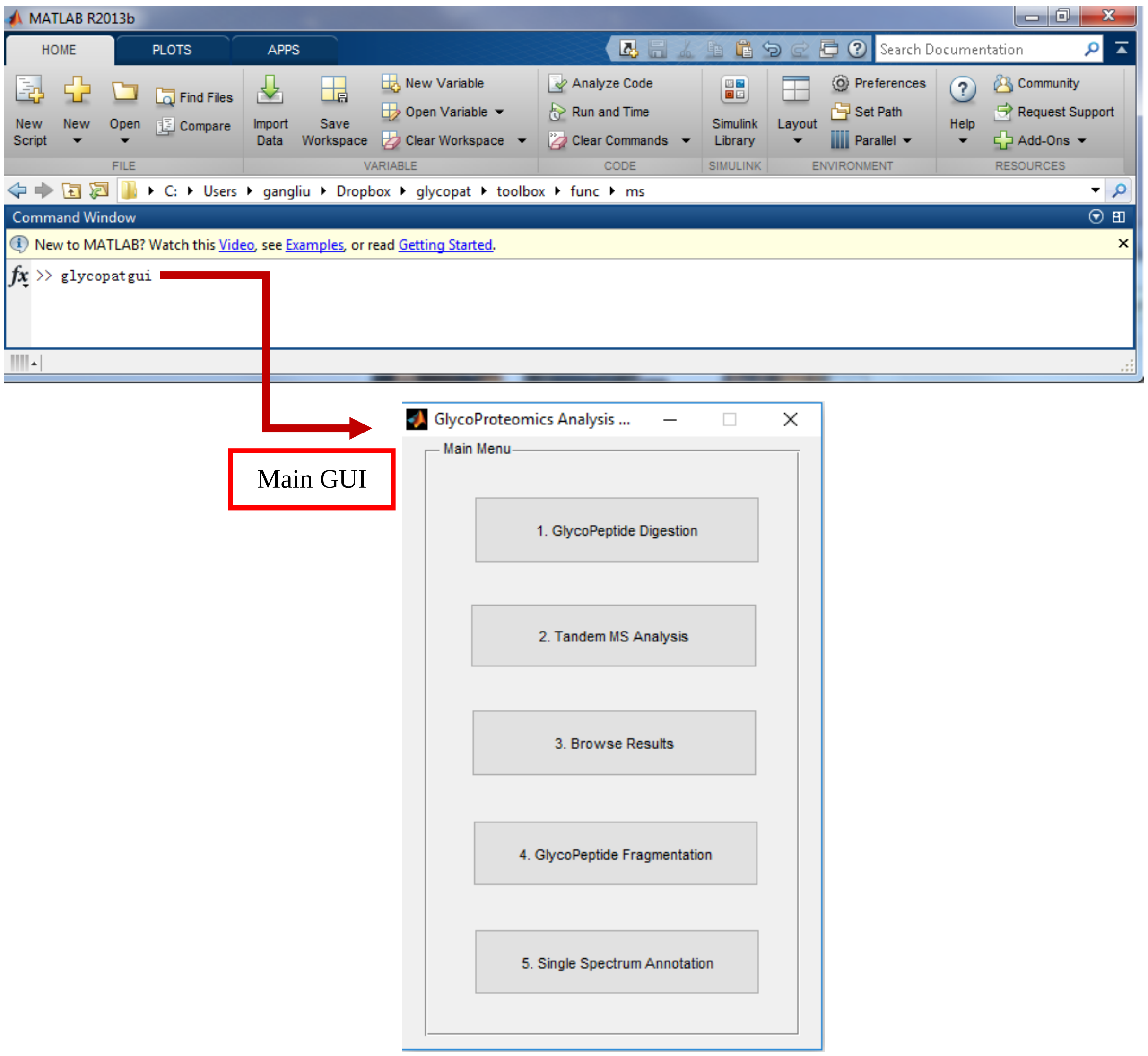
  ```
  help digestSGP
  ```

3. For similar information in the MATLAB help browser, type “`doc <functionName>`”. This displays comment lines written into the GlycoPAT source code.

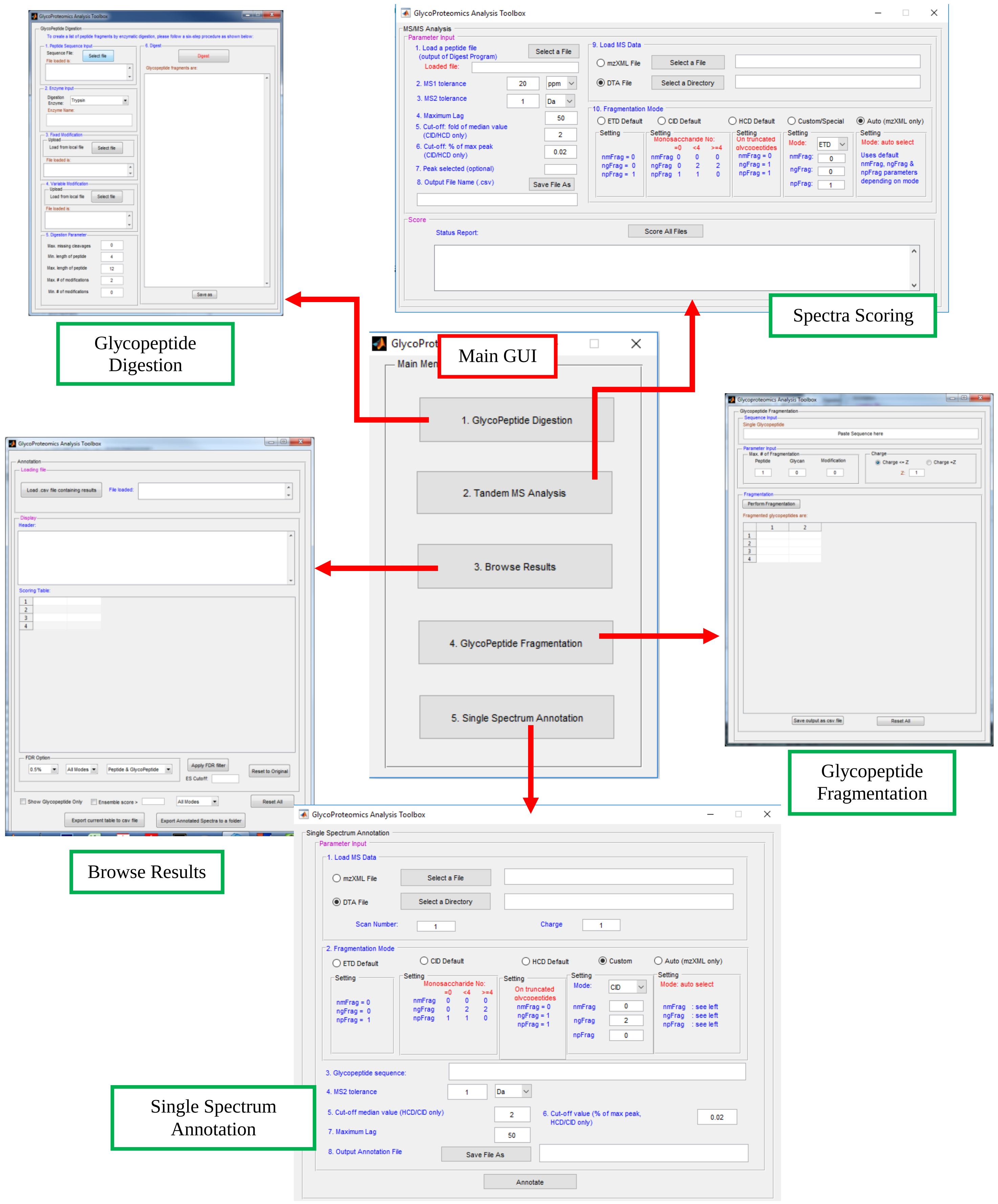
5. Using user-friendly GUIs for GlycoProteomics Analysis: *for people not comfortable with writing MATLAB scripts*

GlycoPAT provides a number of user-friendly GUIs for those who prefer graphical user interfaces (GUIs). To use these facilities simply type "glycopatgui" on the main command prompt (see below). This triggers the "Main GUI". Mouse clicking on the main GUI triggers additional GUIs related to:
"1. GlycoPetide Digestion"
"2. Tandem MS Analysis and Scoring"
"3. Browsing Results",
"4. Theoretical GlycoPeptide Fragmentation", and
"5. Single Spectrum Annotation".
These above GUIs are designed to be intuitive and they perform all necessary steps required for a typical MS/MS Glycoproteomics data analysis. These same GUIs also open up when using the stand-alone version of the software.

The sections 5.1-5.3 below describe the three main GUIs that perform: i. GlycoPeptide Digestion (section 5.1); ii. Tandem MS Analysis (section 5.2); iii. Browse Results (section 5.3). These are the primary GUIs used for data analysis. Additional (optional) GUIs are also available for the theoretical fragmentation of a given glycopeptide, and for the analysis of a single MS^2 Spectrum file. The operations facilitated by these last two GUIs and their usage follow from the examples in section 5.2 and 5.3.



```
>> glycopatgui
```



GlycoProteomics Analysis Toolbox

Main Menu

1. GlycoPeptide Digestion

2. Tandem MS Analysis

3. Browse Results

4. GlycoPeptide Fragmentation

5. Single Spectrum Annotation

GlycoProteomics Analysis Toolbox

GlycoPeptide Digestion

To create a list of peptide fragments by enzymatic digestion, please follow a six-step procedure as shown below:

1. Peptide Sequence Input
Sequence File: Select file
File loaded is:

2. Enzyme Input
Digestion Enzyme: Trypsin
Enzyme Name:

3. Fixed Modification
Upload
Load from local file Select file
File loaded is:

4. Variable Modification
Upload
Load from local file Select file
File loaded is:

5. Digestion Parameter

Max. missing cleavages	0
Min. length of peptide	4
Max. length of peptide	12
Max. # of modifications	2
Min. # of modifications	0

6. Digest
Digest
Glycopeptide fragments are:
Save as

GlycoProteomics Analysis Toolbox

MS/MS Analysis

Parameter Input

1. Load a peptide file (output of Digest Program) Select a File
Loaded file:
2. MS1 tolerance 20 ppm
3. MS2 tolerance 1 Da
4. Maximum Lag 50
5. Cut-off: fold of median value (CID/HCD only) 2
6. Cut-off: % of max peak (CID/HCD only) 0.02
7. Peak selected (optional)
8. Output File Name (.csv) Save File As

9. Load MS Data
mzXML File Select a File
[✓ DTA File] Select a Directory

10. Fragmentation Mode
ETD Default / CID Default / HCD Default / Custom/Special / [✓ Auto (mzXML only)]

ETD Default Setting: nmFrag = 0, ngFrag = 0, npFrag = 1

CID Default Setting: Monosaccharide No:

	=0	<4	>=4
nmFrag	0	0	0
ngFrag	0	2	2
npFrag	1	1	0

HCD Default Setting: On truncated glycopeptides nmFrag = 0, ngFrag = 1, npFrag = 1

Custom/Special Setting: Mode: ETD; nmFrag: 0; ngFrag: 0; npFrag: 1

Auto Setting: Mode: auto select. Uses default nmFrag, ngFrag & npFrag parameters depending on mode

Score
Status Report:
Score All Files

GlycoProteomics Analysis Toolbox

Annotation
Loading file
Load .csv file containing results File loaded:
Display
Header:
Scoring Table:

1
2
3
4

FDR Option 0.5% All Modes Peptide & GlycoPeptide
Apply FDR filter
ES Cutoff:
Reset to Original
Show Glycopeptide Only Ensemble score > All Modes Reset All
Export current table to csv file
Export Annotated Spectra to a folder

Glycoproteomics Analysis Toolbox

Glycopeptide Fragmentation
Sequence Input
Single Glycopeptide
Paste Sequence here
Parameter Input
Max. # of Fragmentation

Peptide	Glycan	Modification
1	0	0

Charge
[✓ Charge <= Z] Charge =Z
Z: 1

Fragmentation
Perform Fragmentation
Fragmented glycopeptides are:

	1	2
1		
2		
3		
4		

Save output as csv file Reset All

GlycoProteomics Analysis Toolbox

Single Spectrum Annotation
Parameter Input

1. Load MS Data
mzXML File Select a File
[✓ DTA File] Select a Directory
Scan Number: 1 Charge 1

2. Fragmentation Mode
ETD Default / CID Default / HCD Default / [✓ Custom] / Auto (mzXML only)

ETD Default Setting: nmFrag = 0, ngFrag = 0, npFrag = 1

CID Default Setting: Monosaccharide No:

	=0	<4	>=4
nmFrag	0	0	0
ngFrag	0	2	2
npFrag	1	1	0

HCD Default Setting: On truncated glycopeptides nmFrag = 0, ngFrag = 1, npFrag = 1

Custom Setting: Mode: CID; nmFrag 0; ngFrag 2; npFrag 0

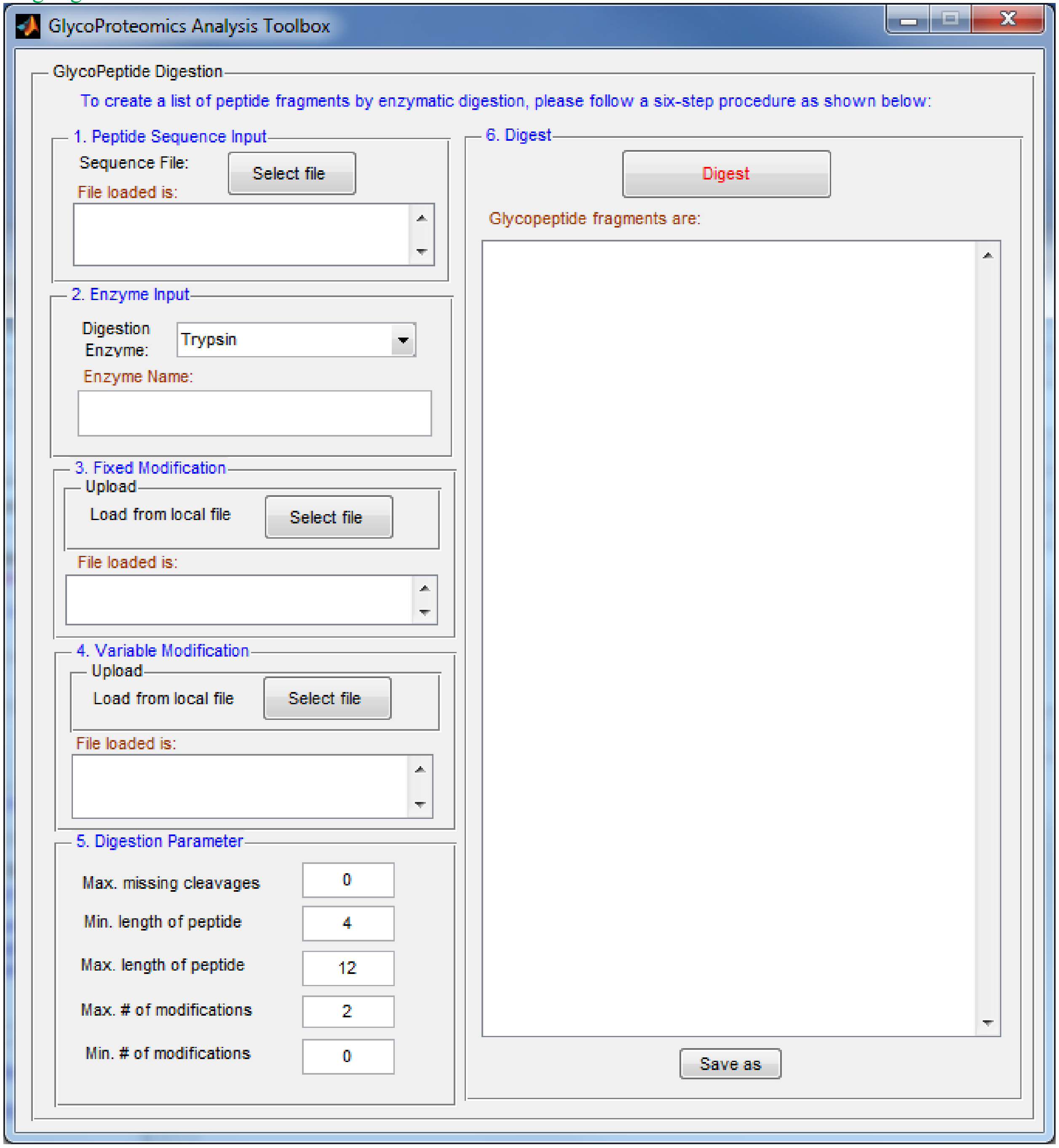
Auto Setting: Mode: auto select. nmFrag : see left, ngFrag : see left, npFrag : see left

3. Glycopeptide sequence:
4. MS2 tolerance 1 Da
5. Cut-off median value (HCD/CID only) 2
6. Cut-off value (% of max peak, HCD/CID only) 0.02
7. Maximum Lag 50
8. Output Annotation File Save File As

Annotate

5.1 Glycopeptide Digestion

Here, the *in silico* glycopeptide library is generated using user-provided protein sequence and protease(s). To initiate this GUI, click "GlycoPeptide Digestion" on the "main GUI". Alternatively, the user can type "digestgui' in the command window.



To create a list of peptide fragments by enzymatic digestion, please follow a six-step procedure as shown below:

Sequence File:

Digestion Enzyme: Trypsin

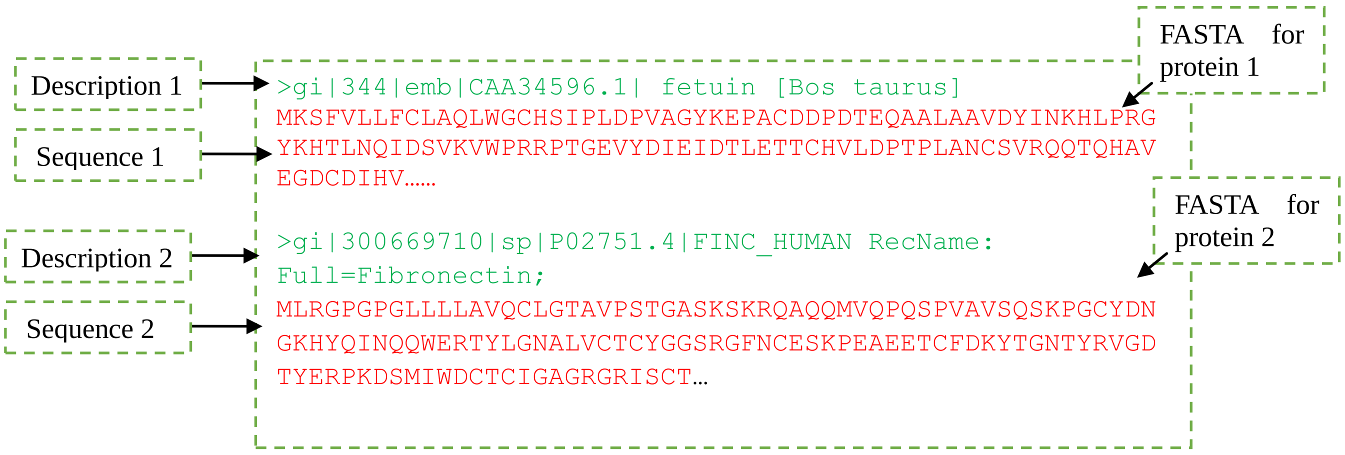
Load from local file

Load from local file

Max. missing cleavages	0
Min. length of peptide	4
Max. length of peptide	12
Max. # of modifications	2
Min. # of modifications	0

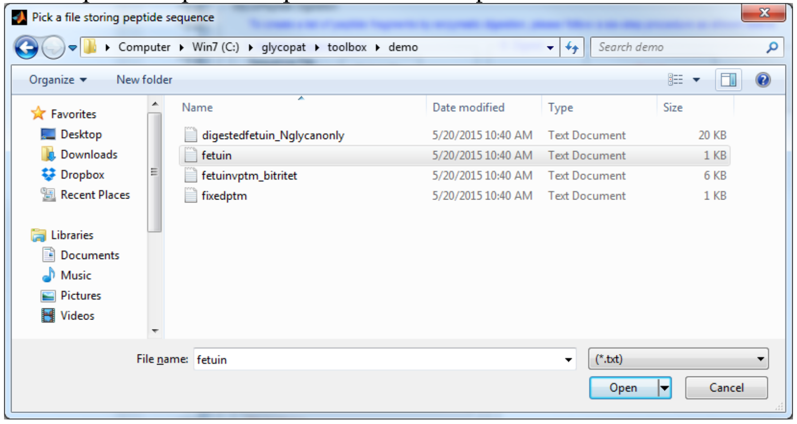
To perform glycopeptide digestion, the user needs to specify the following input files and parameters:

1) **Peptide Sequence file: GlycoPAT** supports the standard FASTA format. Individual protein sequences with their respective headers (initiated by '>') and sequence information can be collate as below:



```
>gi|344|emb|CAA34596.1| fetuin [Bos taurus]
MKSFVLLFCLAQLWGCHSIPLDPVAGYKEPACDDPDTEQAALAAVDYINKHLPRG
YKHTLNQIDSVKVWPRRPTGEVYDIEIDTLETTCHVLDPTPLANCSVRQQTQHAV
EGDCDIHV……

>gi|300669710|sp|P02751.4|FINC_HUMAN RecName:
Full=Fibronectin;
MLRGPGPGLLLLAVQCLGTAVPSTGASKSKRQAQQMVQPQSPVAVSQSKPGCYDN
GKHYQINQQWERTYLGNALVCTCYGGSRGFNCESKPEAEETCFDKYTGNTYRVGD
TYERPKDSMIWDCTCIGAGRGRISCT…
```

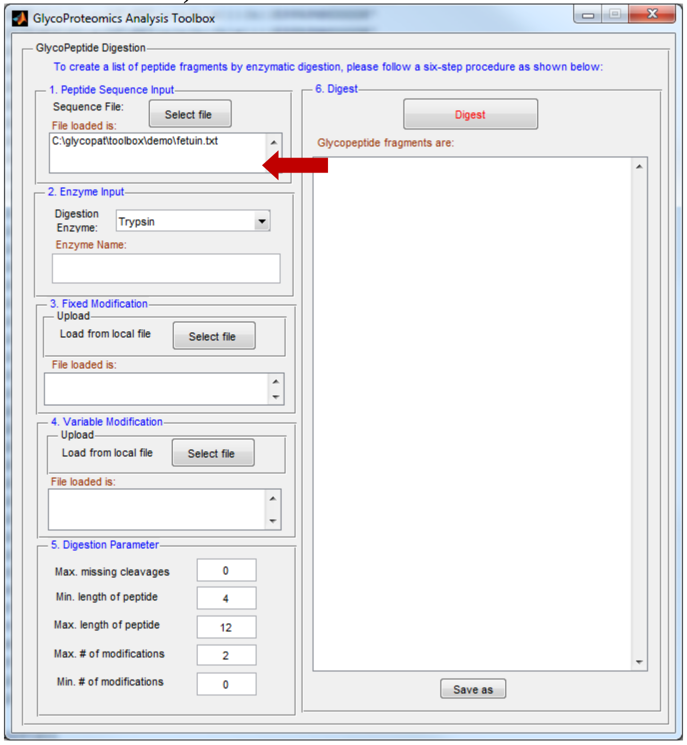
When the user clicks "Select File" Button, a file selection menu will pop up and ask the user to select the 'Peptide Sequence Input' file. For sample file browse to ~toolbox/demo/fetuin.txt



Name	Date modified	Type	Size
digestedfetuin_Nglycanonly	5/20/2015 10:40 AM	Text Document	20 KB
fetuin	5/20/2015 10:40 AM	Text Document	1 KB
fetuinvptm_bitritet	5/20/2015 10:40 AM	Text Document	6 KB
fixedptm	5/20/2015 10:40 AM	Text Document	1 KB

File name: fetuin
(*.txt)

When the file is selected, the name and path of the selected file will display in the status report display (see arrow below).



To create a list of peptide fragments by enzymatic digestion, please follow a six-step procedure as shown below:

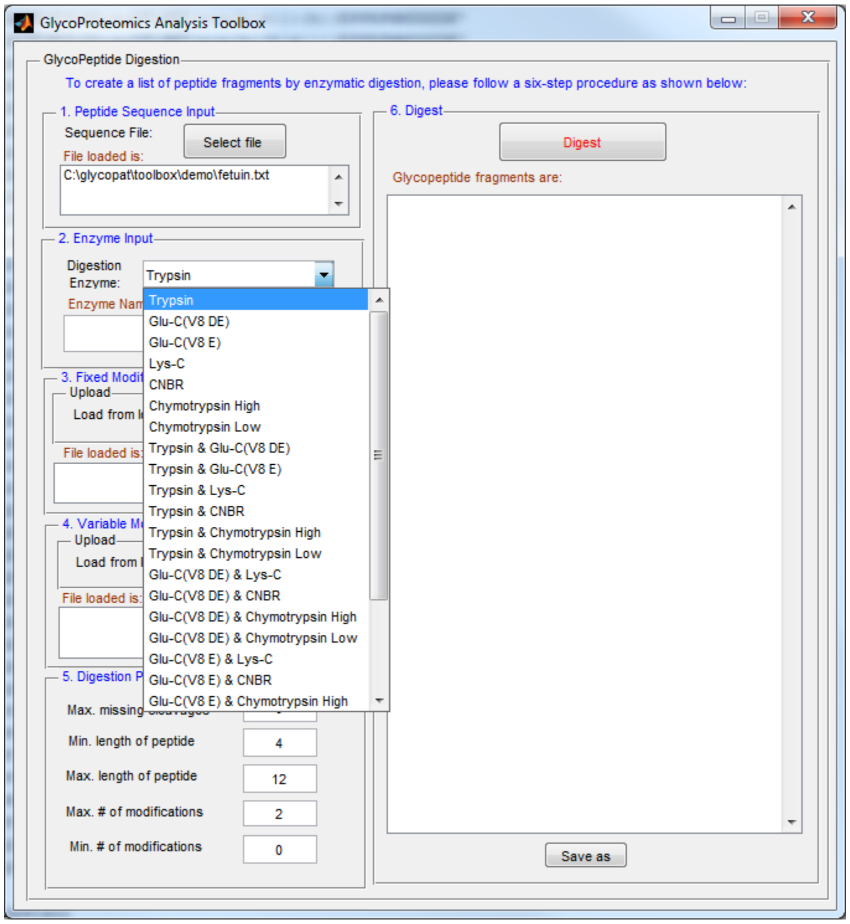
Sequence File:
File loaded is: C:\glycopat\toolbox\demo\fetuin.txt

Digestion Enzyme: Trypsin
Enzyme Name:

Max. missing cleavages	0
Min. length of peptide	4
Max. length of peptide	12
Max. # of modifications	2
Min. # of modifications	0

2)Enzyme Input:

The user can select the enzyme from the pull-down menu in the GUI as shown below.



To create a list of peptide fragments by enzymatic digestion, please follow a six-step procedure as shown below:

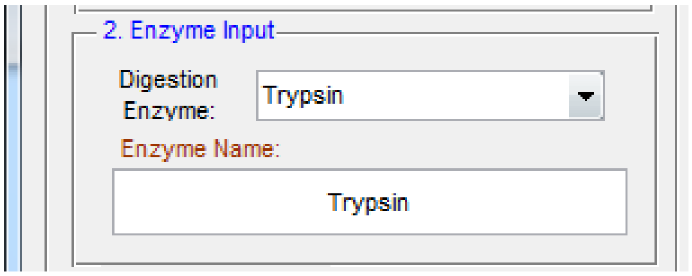
Sequence File:
File loaded is: C:\glycopat\toolbox\demo\fetuin.txt

Digestion Enzyme: Trypsin

Trypsin
Glu-C(V8 DE)
Glu-C(V8 E)
Lys-C
CNBR
Chymotrypsin High
Chymotrypsin Low
Trypsin & Glu-C(V8 DE)
Trypsin & Glu-C(V8 E)
Trypsin & Lys-C
Trypsin & CNBR
Trypsin & Chymotrypsin High
Trypsin & Chymotrypsin Low
Glu-C(V8 DE) & Lys-C
Glu-C(V8 DE) & CNBR
Glu-C(V8 DE) & Chymotrypsin High
Glu-C(V8 DE) & Chymotrypsin Low
Glu-C(V8 E) & Lys-C
Glu-C(V8 E) & CNBR
Glu-C(V8 E) & Chymotrypsin High

Max. missing cleavages	
Min. length of peptide	4
Max. length of peptide	12
Max. # of modifications	2
Min. # of modifications	0

After the user selects the protease, the enzyme name will display in the text box as below.



Digestion Enzyme: Trypsin

Enzyme Name: Trypsin

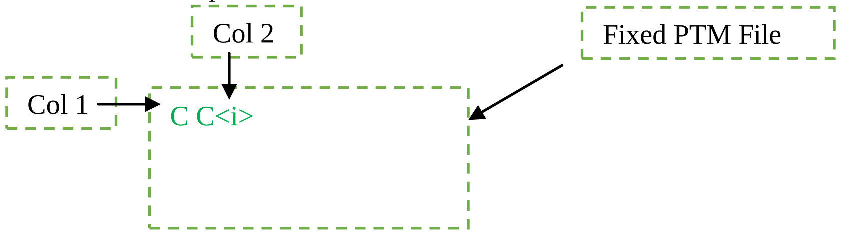
Currently, the "**GlycoPeptide Digestion**" GUI supports five proteases: trypsin, Glu-C, Lys-C, CNBR, and Chymotrypsin. Menu selection options include different digestion rules with the same enzyme since the specificity can vary with digestion conditions. For example, two digestion rules are allowed for Glu-C including digestion at either E (glutamic acid) alone, or both D (Aspartic acid) and E sites. Options for multiple enzyme digestion are also available.

In addition to the limited set of proteases available with the GUI, **GlycoPAT** also includes a local protease database consisting of 40 additional digestion rules and associated enzymes. The data are identical to the specificity defined in the ExPASY **PeptideCutter** (http://web.expasy.org/peptide_cutter/peptidecutter_enzymes.html).

To view the enzyme rules available in this database, please type "help Protease" in the MATLAB command window. To perform protein digestions using these enzymes, write MATLAB scripts (example provided in section 6.3).

3)Fixed Modification File:

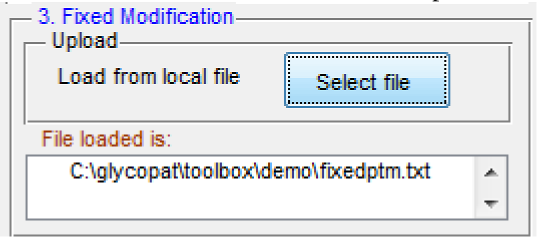
Fixed modifications are non-glycan modifications that can occur due to chemical derivation of peptides, e.g. Carbamidomethylation of Cys residues. These are described in the 'Fixed Modification File' in plain text format. This file contains two space-delimited columns, with the first column listing the amino acid being modified. The second column lists the amino acid name followed by modification enclosed within "<" and ">". The modification itself is represented by a single letter as shown in the example below:



GlycoPAT supports 11 types of potential fixed modifications. Symbols used to represent these are listed below:

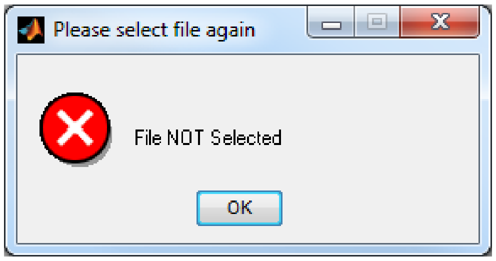
Modification	*Symbol*
Oxidation	o
Sulfation	s
Carbamidomethylation	i
Carboxymethylation	c
Phosphorylation	p
Acetylation	a
Biotinylation	b
Pyro-Glu Q	q
Pyro-Glu E	e
Formylation	f
Sodiation	n

Similar to the 'Peptide Sequence Input' file, the user can click the 'Select File' button and load the fixed modification file into the GUI. The file information shows up in the status panel:



C:\glycopat\toolbox\demo\fixedptm.txt

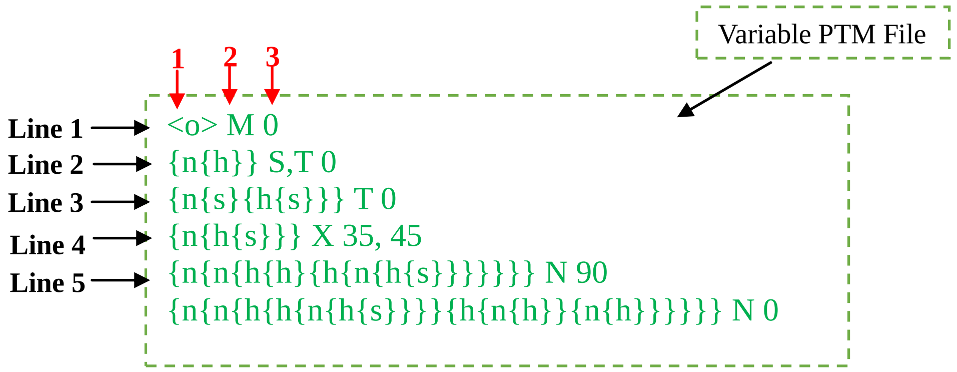
If the file is not loaded, the GUI will notify the user that the file is not selected and ask the user to select file again.


Please select file again

File NOT Selected

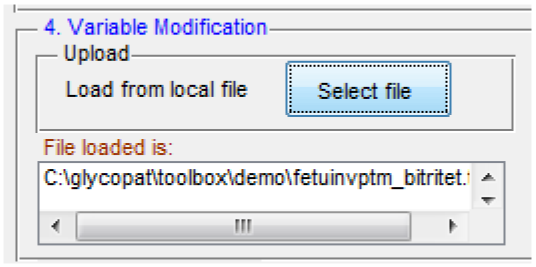
4) Variable Modification File:

The ‘variable modification file’ is a text file with three space-delimited columns (see example below). The first column has the name of the modification, which can include both non-glycan (line 1, below) and glycan (line 2-5, below) modifications. For glycan modifications, the “SmallGlyPep” format is used (see main manuscript). The second column names one or more amino acid(s) separated by comma(s) that is being modified. For example, oxidation can occur on Met (line 1) and the ‘{n{h}}’ modification can occur on either Ser or Thr (line 2). If 'X' is placed on the second column, the variable modification does not take place on any specific amino acid. In this case, the third column may specify specific amino acids where the variable modification may occur (e.g. at amino acids 35 and 45 on line 4). Indeed, variable modifications can be specified to occur both at specific amino acids and specific position, as shown in line 5. Further, N-glycans specified to occur at Asp (N) residues (line 5 and 6) only occurs at Asp sites provided it lies in the N-X-S/T consensus sequon.



```
<o> M 0
{n{h}} S,T 0
{n{s}{h{s}}} T 0
{n{h{s}}} X 35, 45
{n{n{h{h}{h{n{h{s}}}}}}} N 90
{n{n{h{h{n{h{s}}}}{h{n{h}}{n{h}}}}}} N 0
```

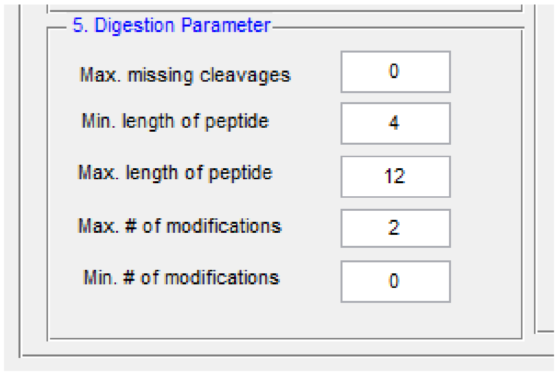
Similar to the ‘Fixed Modification file’ input, the user can upload the ‘Variable Modification file’ into the GUI. Once loaded, the file name and path are displayed in the status panel below:



C:\glycopat\toolbox\demo\fetuinvptm_bitritet.

5)Digestion Parameter

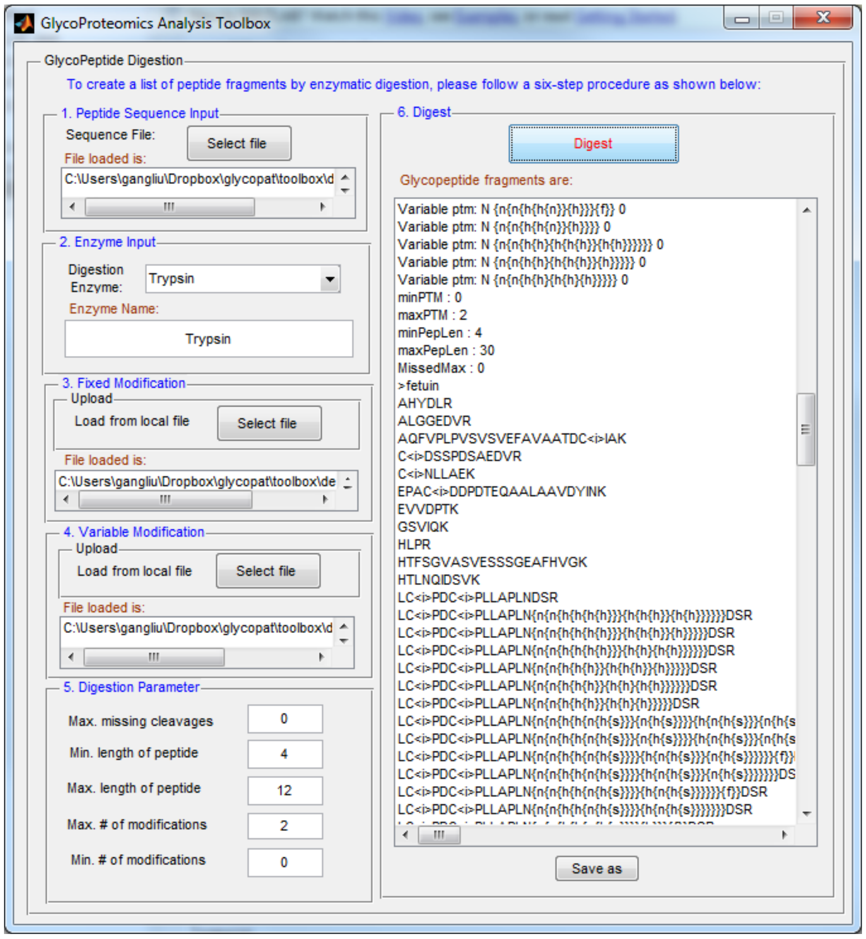
Several additional user-specified parameters can be input using the GUI as shown below. These include the maximum number of missing cleavage sites, minimum and maximum length of digested peptide, and minimum and maximum number of variable modifications in the digested peptide. The default values for these parameters are provided in the figure below. In this regard, we specifically allow users to limit the number of variable modifications in order to limit the size of the theoretical glycopeptide library generated by the computer program.


5. Digestion Parameter

Max. missing cleavages	0
Min. length of peptide	4
Max. length of peptide	12
Max. # of modifications	2
Min. # of modifications	0

6)Digestion

Clicking the "Digest" button at the top of the right panel initiates the GlycoPeptide Digestion calculations. Upon completion of the computations, all the possible theoretical glycopeptides resulting from the enzymatic digestion show up in the "Digest" Panel as shown below.


GlycoProteomics Analysis Toolbox

GlycoPeptide Digestion

To create a list of peptide fragments by enzymatic digestion, please follow a six-step procedure as shown below:

1. Peptide Sequence Input
Sequence File: Select file
File loaded is:
C:\Users\gangliu\Dropbox\glycopat\toolbox\d

2. Enzyme Input
Digestion Enzyme: Trypsin
Enzyme Name:
Trypsin

3. Fixed Modification
Upload
Load from local file Select file
File loaded is:
C:\Users\gangliu\Dropbox\glycopat\toolbox\de

4. Variable Modification
Upload
Load from local file Select file
File loaded is:
C:\Users\gangliu\Dropbox\glycopat\toolbox\d

5. Digestion Parameter

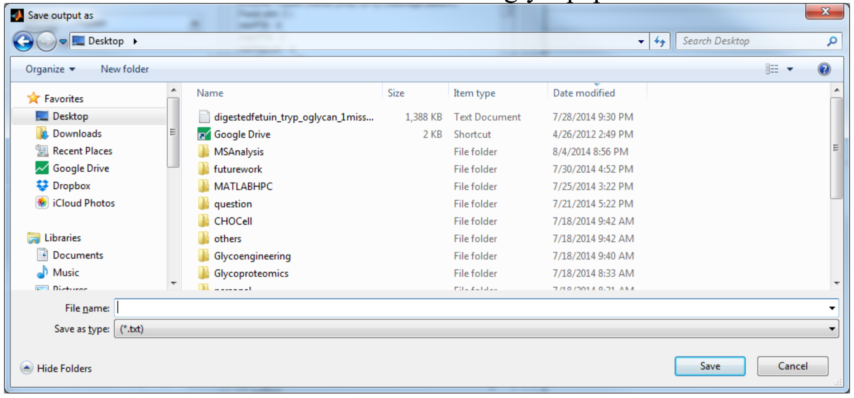
Max. missing cleavages	0
Min. length of peptide	4
Max. length of peptide	12
Max. # of modifications	2
Min. # of modifications	0

6. Digest
Digest
Glycopeptide fragments are:

```
Variable ptm: N {n{n{h{h{n}}{h}}}{f}} 0
Variable ptm: N {n{n{h{h{n}}{h}}}} 0
Variable ptm: N {n{n{h{h}{h{h{h}}{h{h}}}}}} 0
Variable ptm: N {n{n{h{h}{h{h{h}}{h}}}}} 0
Variable ptm: N {n{n{h{h}{h{h}{h}}}}} 0
minPTM : 0
maxPTM : 2
minPepLen : 4
maxPepLen : 30
MissedMax : 0
>fetuin
AHYDLR
ALGGEDVR
AQFVPLPVSVSVEFAVAATDC<i>IAK
C<i>DSSPDSAEDVR
C<i>NLLAEK
EPAC<i>DDPDTEQAALAAVDYINK
EVVDPTK
GSVIQK
HLPR
HTFSGVASVESSSGEAFHVGK
HTLNQIDSVK
LC<i>PDC<i>PLLAPLNDSR
LC<i>PDC<i>PLLAPLN{n{n{h{h{h{h}}}{h{h{h}}{h{h}}}}}}DSR
LC<i>PDC<i>PLLAPLN{n{n{h{h{h{h}}}{h{h{h}}{h}}}}}DSR
LC<i>PDC<i>PLLAPLN{n{n{h{h{h{h}}}{h{h}{h{h}}}}}}DSR
LC<i>PDC<i>PLLAPLN{n{n{h{h{h}}{h{h{h}}{h}}}}}DSR
LC<i>PDC<i>PLLAPLN{n{n{h{h{h}}{h{h}{h{h}}}}}}DSR
LC<i>PDC<i>PLLAPLN{n{n{h{h{h}}{h{h}{h}}}}}DSR
LC<i>PDC<i>PLLAPLN{n{n{h{h{n{h{s}}}{n{h{s}}}}{h{n{h{s}}}{n{h{s
LC<i>PDC<i>PLLAPLN{n{n{h{h{n{h{s}}}{n{h{s}}}}{h{n{h{s}}}{n{h{s
LC<i>PDC<i>PLLAPLN{n{n{h{h{n{h{s}}}}{h{n{h{s}}}{n{h{s}}}}}}{f}}
LC<i>PDC<i>PLLAPLN{n{n{h{h{n{h{s}}}}{h{n{h{s}}}{n{h{s}}}}}}}DS
LC<i>PDC<i>PLLAPLN{n{n{h{h{n{h{s}}}}{h{n{h{s}}}}}}{f}}DSR
LC<i>PDC<i>PLLAPLN{n{n{h{h{n{h{s}}}}{h{n{h{s}}}}}}}DSR
```

Save as

The user can click "Save as" button to save the list of glycopeptides as a text file.


Save output as

Desktop ▸ Search Desktop

Organize ▾ New folder

Favorites
Desktop
Downloads
Recent Places
Google Drive
Dropbox
iCloud Photos

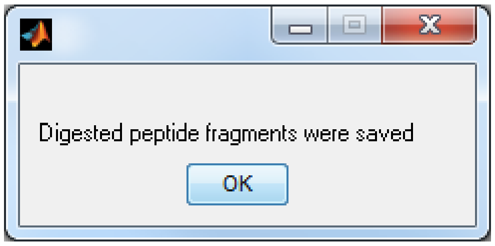
Libraries
Documents
Music

Name	Size	Item type	Date modified
digestedfetuin_tryp_oglycan_1miss...	1,388 KB	Text Document	7/28/2014 9:30 PM
Google Drive	2 KB	Shortcut	4/26/2012 2:49 PM
MSAnalysis		File folder	8/4/2014 8:56 PM
futurework		File folder	7/30/2014 4:52 PM
MATLABHPC		File folder	7/25/2014 3:22 PM
question		File folder	7/21/2014 5:22 PM
CHOCell		File folder	7/18/2014 9:42 AM
others		File folder	7/18/2014 9:42 AM
Glycoengineering		File folder	7/18/2014 9:40 AM
Glycoproteomics		File folder	7/18/2014 8:33 AM

File name:
Save as type: (*.txt)

Hide Folders Save Cancel

After the peptide file is saved, the user will be notified and the saved file can be used for tandem MS data analysis.


Digested peptide fragments were saved

OK

5.2 Tandem MS Analysis and Scoring

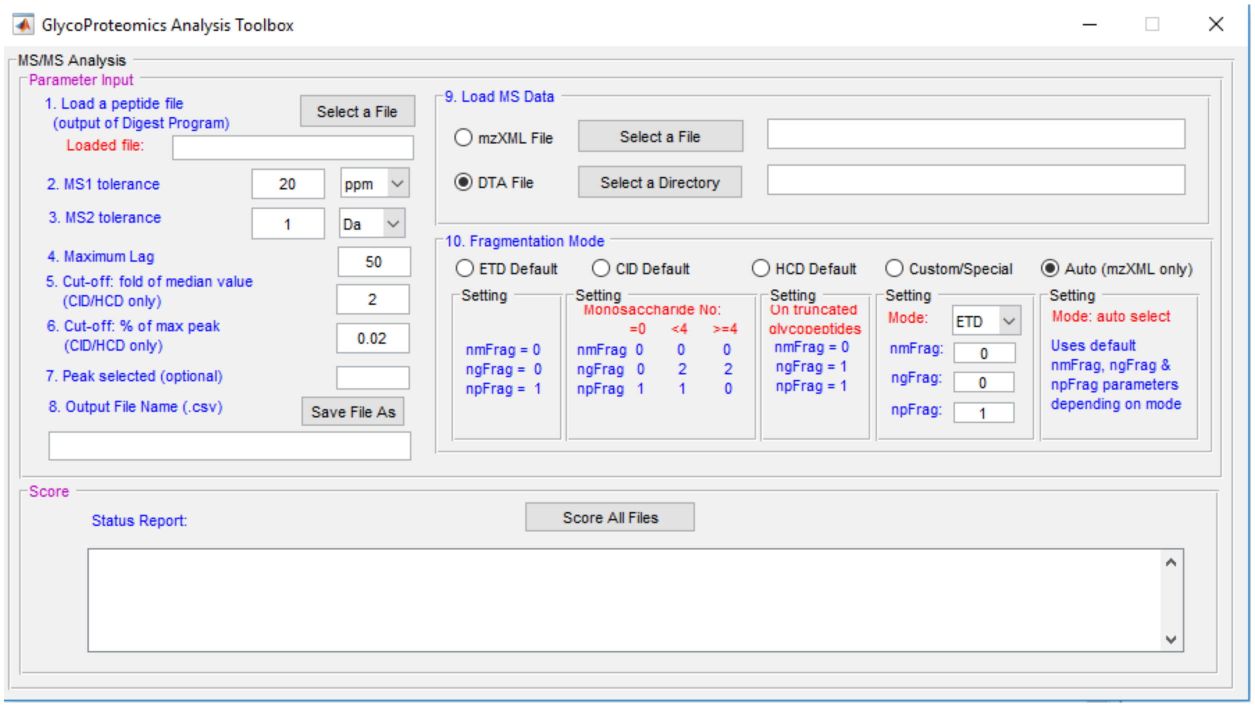
Following generation of the glycopeptide digest library (section 5.1), the user can initiate 'Tandem MS Analysis'. This involves two sequential steps for each MS^2 spectrum:

STEP 1: The experimental mass-to-charge (*m/z*) ratio of the precursor ion of the MS^2 spectrum are compared against the theoretical glycopeptide digest library to identify the putative list of 'candidate glycopeptides' that may explain the spectrum. The mass tolerance for such MS^1 matching is user-specified.

STEP 2: For each 'candidate glycopeptide', a scoring algorithm is implemented to calculate the *Ensemble Score* (ES). This ES evaluates the likelihood that the theoretical MS^2 spectrum of the 'candidate glycopeptide' matches the experimental spectrum. This score is based on four parameters:

- Cross correlation analysis parameter, *Xcorr*
- Probability-based *P-value* estimated using experimental data and corresponding glycopeptide decoy database
- % of theoretical fragmentation ions matched in experimental spectrum
- Fraction of Top 10 peak experimental peaks that are identical to the predicted theoretical peaks.

To start "Spectra scoring" GUI, the user can click "Tandem MS Analysis" button in the main GUI or type "scoregui" a the command window.


GlycoProteomics Analysis Toolbox

MS/MS Analysis

Parameter Input

1. Load a peptide file (output of Digest Program) — Select a File

Loaded file:

2. MS1 tolerance: 20 ppm

3. MS2 tolerance: 1 Da

4. Maximum Lag: 50

5. Cut-off: fold of median value (CID/HCD only): 2

6. Cut-off: % of max peak (CID/HCD only): 0.02

7. Peak selected (optional):

8. Output File Name (.csv) — Save File As

9. Load MS Data

mzXML File — Select a File [✓ DTA File] — Select a Directory

10. Fragmentation Mode

ETD Default — Setting: nmFrag = 0, ngFrag = 0, npFrag = 1

CID Default — Setting: Monosaccharide No:

	=0	<4	>=4
nmFrag	0	0	0
ngFrag	0	2	2
npFrag	1	1	0

HCD Default — Setting: On truncated glycopeptides: nmFrag = 0, ngFrag = 1, npFrag = 1

Custom/Special — Setting: Mode: ETD; nmFrag: 0; ngFrag: 0; npFrag: 1

[✓ Auto (mzXML only)] — Setting: Mode: auto select. Uses default nmFrag, ngFrag & npFrag parameters depending on mode

Score

Status Report: — Score All Files

When the GUI appears, the user has to provide 9 parameters listed below:

1) **Peptide file**: click "Select a File" button to select the output from the "GlycoPeptide Digestion" step (section 5.1). This file specifies the 'glycopeptide digest library'.
2) **MS1 Tolerance**: This specifies the MS^1 tolerance value and unit used in STEP 1 above. The default MS1 tolerance value is 20 and MS^1 unit is 'ppm' (see above).
3) **MS2 Tolerance**: MS^2 tolerance value and unit are used for STEP 2 above. The default MS^2 tolerance value is 1 and MS^2 unit is 'Da'.

4) **Maximum lag:** Maximum number of spectra shifting units used for Xcorr calculation. Default value is 50 (see main manuscript).

5) **Cut-off median value and % of maximum peak:** MS^2 spectrum often have some baseline signal that have low signal/noise (S/N) ratio. These do not contribute to MS^2 spectrum matching. They can be removed by considering the median signal of the spectrum (which is usually equivalent to the instrument noise) in comparison to the signal of the most intense peak. By default, GlycoPAT removes all peaks that are less than two-times the median signal in the spectrum (i.e. 'Cut-off: fold of median value'=2) provided these signals are less than 2% of the most intense peak (i.e. 'Cut-off: % max peak'=0.02). These parameters can be further tuned by user if necessary. Note that these cut-off parameters only apply to studies using CID/HCD mode fragmentation. Local noise reduction is applied in the ETD mode, where: 1. The precursor ion peak is automatically removed; and 2. Peaks below the median value in local 100Th m/z window are deleted.

6) **Peak selected (optional):** This parameter does not affect the *ES* scoring scheme. It only identifies spectrum with specific glycan 'signature peaks' in the output file. For example, if one is interested in determining if a given matched spectrum has sialic acid (Neu5Ac) and HexHexNAc residues, the 'Peak selected' field may be populated '291, 366' corresponding to the expected masses. The output file then specifies if these specific peaks were identified in a given spectrum.

7) **Output File Name:** The user can click on the "Save File As" button to select the output location and file name. Final results are saved in a comma separated variable (CSV) file that can be opened using either excel or the 'Browse Results' GUI (section 5.3). Selected path and file name will be shown in the display status panel.

8) **Load MS Data:** GlycoPAT provides the user two options to load experimental LC/MS data. The user can use freeware like *ProteomicsWizard* to convert the MS data to mzXML format. If this is available, the user can choose the "mzXML File" radio button in the GUI and use "Select a File" to choose the mzXML file for analysis.

   Alternatively, the user can generate text/raw data files in DTA format using software like Thermo *BioWorks*. To load these data, click the radio button next to "DTA File", and specify storage directory by choosing "Select a Directory".

   The user-specified file/data location is shown in the status report panel.

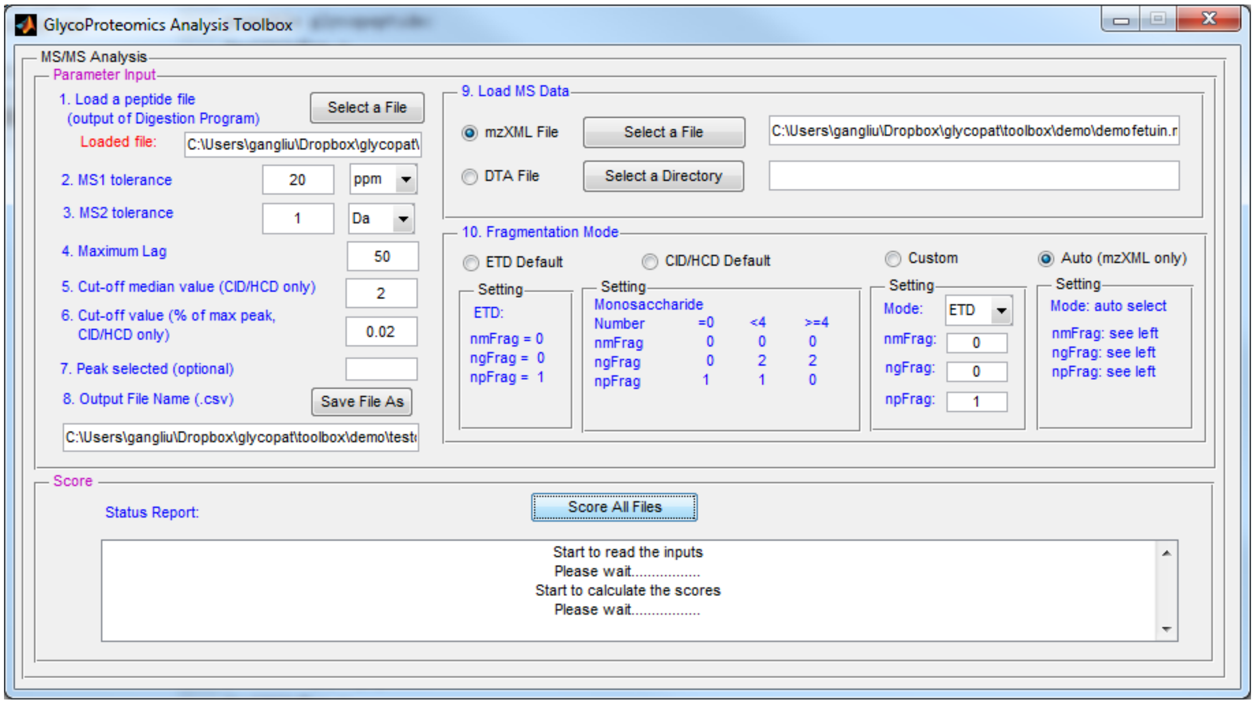
9) **Fragmentation Mode**: The selection of fragmentation mode is dependent on the specific method used for MS experiments. Five options are available that can be selected by choosing the appropriate radio buttons:

- The 'ETD default' setting assumes that the (glyco)peptide is fragmented once into *c* and *z* ions regardless of the presence of glycans (see figure below, `npFrag` =1).
- The 'CID Default' setting assumes that peptides are fragmented once into *y* and *z* ions if there are no glycans attached (i.e. `npFrag`=1 if `Monosaccharide count`=0); multiple fragmentation can occur on both the glycan and peptide if small glycans with less than 4 monosaccharides are attached to the glycopeptide (i.e. `ngFrag` =2, `npFrag` =1 if `Monosaccharide count` <4); and up to two glycan fragmentations can occur for larger glycopeptides since this is a common occurrence in peptides bearing N-linked glycans (i.e. `ngFrag` =2 if `Monosaccharide count` $\geq$4).
- The 'HCD Default' setting assumes that prominent Y0 (peptide alone) and Y1 (peptide with 1 monosaccharide) ions commonly occur when glycopeptides are fragmented in high-energy HCD mode, since the glycans are otherwise shattered. Additionally, peptide fragmentation can occur in this mode, only the glycan is truncated with small carbohydrate structures containing only 1 or 2 glycosidic bond linkages. Due to this, fragmentation of N-linked glycopeptides is thought to result in truncated glycopeptides containing HexNAc($\pm$Fuc) and,HexNAc-HexNAc($\pm$Fuc) type stub glycans. Similar small truncated glycopeptides are expected in the case of O-glycans.
- 'Custom' options can also be specified in order to test additional fragmentation patterns in ETD, CID or HCD modes. Here, the specification of 'ETD' mode results in *c* and *z* ions

only, whereas ‘CID’/’HCD’ results in the analysis of product *b* and *y* ions. In this case, the user can specify the number of fragmentations in the non-glycan PTM modifications (`nmFrag`), glycan PTM modification (`ngFrag`) and peptide backbone (`npFrag`). Again, in the case of HCD, the fragmentations suggested in nmFrag, ngFrag and npFrag only occur on truncated glycopeptides bearing small glycan stubs.

- The ‘Auto (mxXML only)’ option is the preferred method if the input data file is an mzXML file. In this case, the program parses the mzXML file to automatically discover the experimental fragmentation mode. It then uses the default settings for ETD, CID or HCD as specified above.

When all the inputs described above are provided, the user can start the scoring process by clicking “Score All File” in the bottom panel. A wait bar will automatically appear to monitor the computing progress as below.



GlycoProteomics Analysis Toolbox

MS/MS Analysis

Parameter Input

1. Load a peptide file (output of Digestion Program) — Select a File

Loaded file: C:\Users\gangliu\Dropbox\glycopat\

2. MS1 tolerance: 20 ppm

3. MS2 tolerance: 1 Da

4. Maximum Lag: 50

5. Cut-off median value (CID/HCD only): 2

6. Cut-off value (% of max peak, CID/HCD only): 0.02

7. Peak selected (optional):

8. Output File Name (.csv) — Save File As

C:\Users\gangliu\Dropbox\glycopat\toolbox\demo\test

9. Load MS Data

[✓ mzXML File] — Select a File — C:\Users\gangliu\Dropbox\glycopat\toolbox\demo\demofetuin.r

DTA File — Select a Directory

10. Fragmentation Mode

ETD Default — Setting: ETD: nmFrag = 0, ngFrag = 0, npFrag = 1

CID/HCD Default — Setting:

Monosaccharide Number	=0	<4	>=4
nmFrag	0	0	0
ngFrag	0	2	2
npFrag	1	1	0

Custom — Setting: Mode: ETD; nmFrag: 0; ngFrag: 0; npFrag: 1

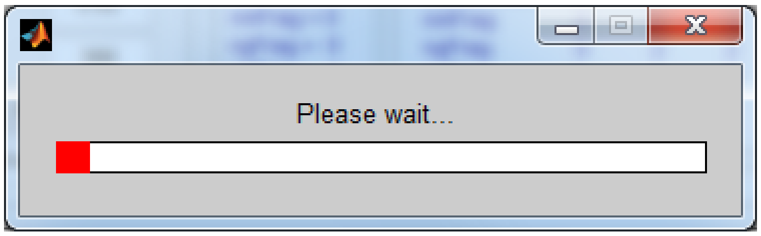
[✓ Auto (mzXML only)] — Setting: Mode: auto select; nmFrag: see left; ngFrag: see left; npFrag: see left

Score

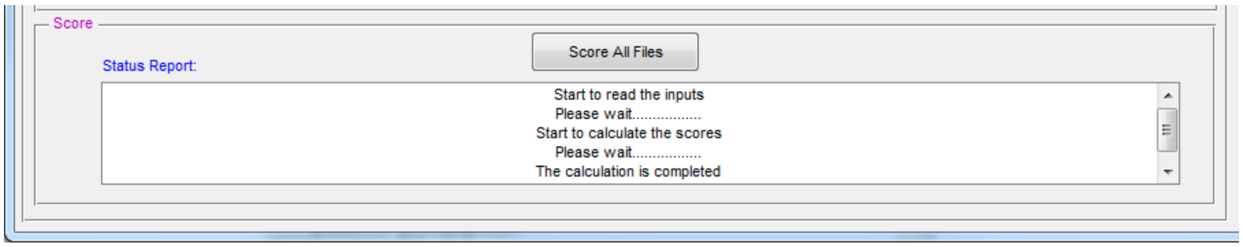
Status Report: — Score All Files

```
Start to read the inputs
Please wait................
Start to calculate the scores
Please wait................
```

At the same time, the computing status report will display in the “Status Report” window in the bottom panel of the GUI. When the computation is complete, the wait bar will close automatically and a message “The calculation is completed” will be shown in the “Status Report” panel.



Please wait...



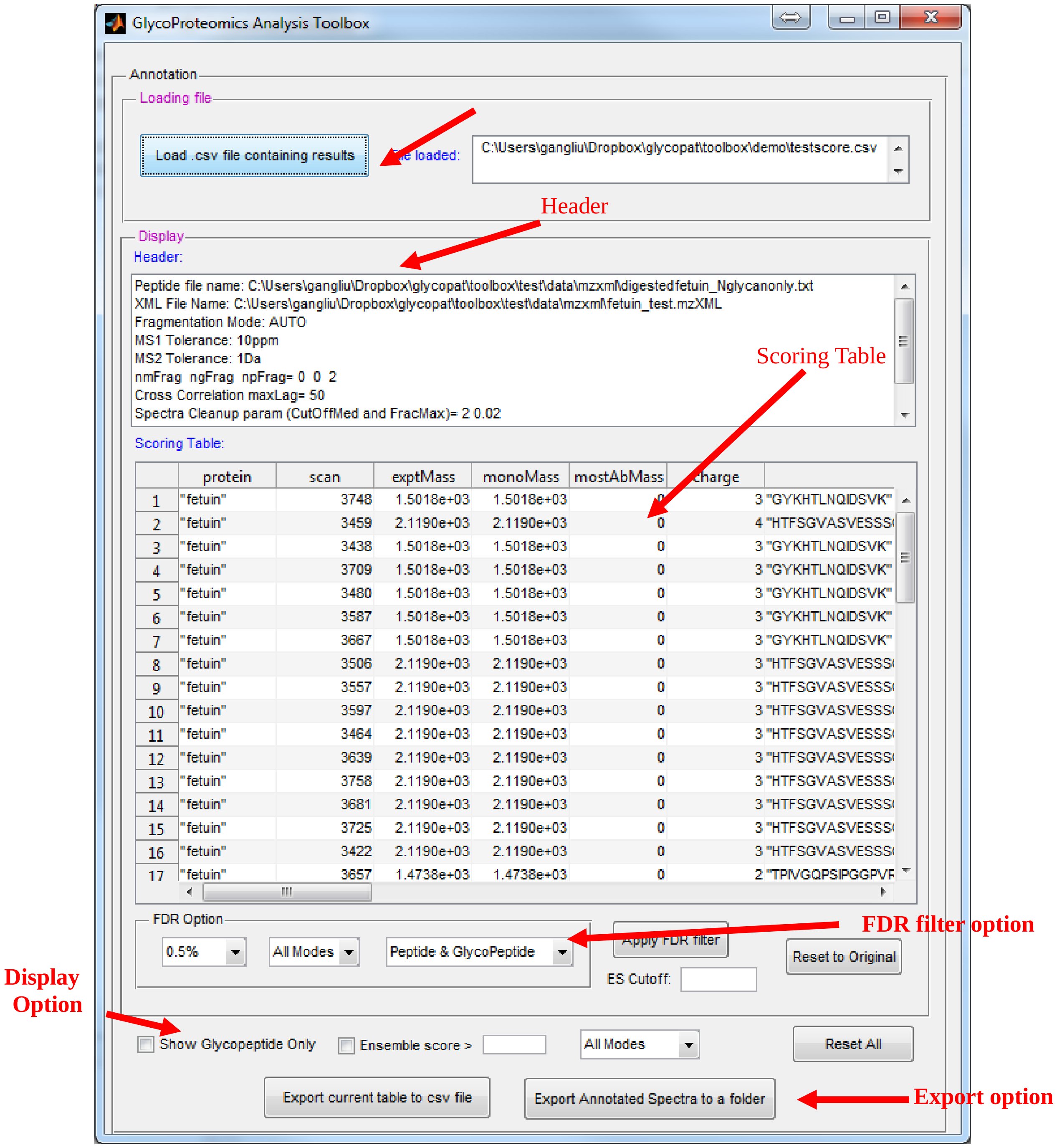
```
Start to read the inputs
Please wait................
Start to calculate the scores
Please wait................
The calculation is completed
```

If the MATLAB command window is open during the calculations, the user will be able to follow the progress as the individual spectrum are analyzed.

The computational time for this step varies depending on the size of the GlycoPeptide digest library and the MS data file. IF THE INPUT FILES ARE LARGE, THE USER MAY HAVE TO WAIT FOR SEVERAL MINUTES BEFORE THE ‘PLEASE WAIT…’ BAR APPEARS AS THE LARGE FILE IS LOADED INTO MEMORY AND THE DATA ANALYSIS IS COMPLETED.

5.3 Browse Results

Upon completing the 'Tandem MS Analysis', the user can view the result by selecting the "Browse Result" option in the main GUI. Alternatively, type "browsegui" in the command window. In the new GUI (shown below), specify the output file from the 'Tandem MS Analysis' by choosing "Load .csv file containing results" (arrow). Similar to the previous GUIs, the path and file name of the input file appears in the status report panel.



C:\Users\gangliu\Dropbox\glycopat\toolbox\demo\testscore.csv

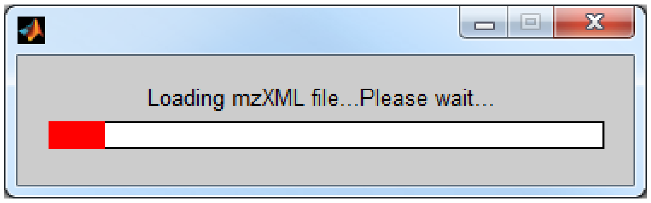
```
Peptide file name: C:\Users\gangliu\Dropbox\glycopat\toolbox\test\data\mzxml\digestedfetuin_Nglycanonly.txt
XML File Name: C:\Users\gangliu\Dropbox\glycopat\toolbox\test\data\mzxml\fetuin_test.mzXML
Fragmentation Mode: AUTO
MS1 Tolerance: 10ppm
MS2 Tolerance: 1Da
nmFrag  ngFrag  npFrag= 0  0  2
Cross Correlation maxLag= 50
Spectra Cleanup param (CutOffMed and FracMax)= 2 0.02
```

	protein	scan	exptMass	monoMass	mostAbMass	charge	
1	"fetuin"	3748	1.5018e+03	1.5018e+03	0	3	"GYKHTLNQIDSVK"
2	"fetuin"	3459	2.1190e+03	2.1190e+03	0	4	"HTFSGVASVESSS
3	"fetuin"	3438	1.5018e+03	1.5018e+03	0	3	"GYKHTLNQIDSVK"
4	"fetuin"	3709	1.5018e+03	1.5018e+03	0	3	"GYKHTLNQIDSVK"
5	"fetuin"	3480	1.5018e+03	1.5018e+03	0	3	"GYKHTLNQIDSVK"
6	"fetuin"	3587	1.5018e+03	1.5018e+03	0	3	"GYKHTLNQIDSVK"
7	"fetuin"	3667	1.5018e+03	1.5018e+03	0	3	"GYKHTLNQIDSVK"
8	"fetuin"	3506	2.1190e+03	2.1190e+03	0	3	"HTFSGVASVESSS
9	"fetuin"	3557	2.1190e+03	2.1190e+03	0	3	"HTFSGVASVESSS
10	"fetuin"	3597	2.1190e+03	2.1190e+03	0	3	"HTFSGVASVESSS
11	"fetuin"	3464	2.1190e+03	2.1190e+03	0	3	"HTFSGVASVESSS
12	"fetuin"	3639	2.1190e+03	2.1190e+03	0	3	"HTFSGVASVESSS
13	"fetuin"	3758	2.1190e+03	2.1190e+03	0	3	"HTFSGVASVESSS
14	"fetuin"	3681	2.1190e+03	2.1190e+03	0	3	"HTFSGVASVESSS
15	"fetuin"	3725	2.1190e+03	2.1190e+03	0	3	"HTFSGVASVESSS
16	"fetuin"	3422	2.1190e+03	2.1190e+03	0	3	"HTFSGVASVESSS
17	"fetuin"	3657	1.4738e+03	1.4738e+03	0	2	"TPIVGQPSIPGGPV

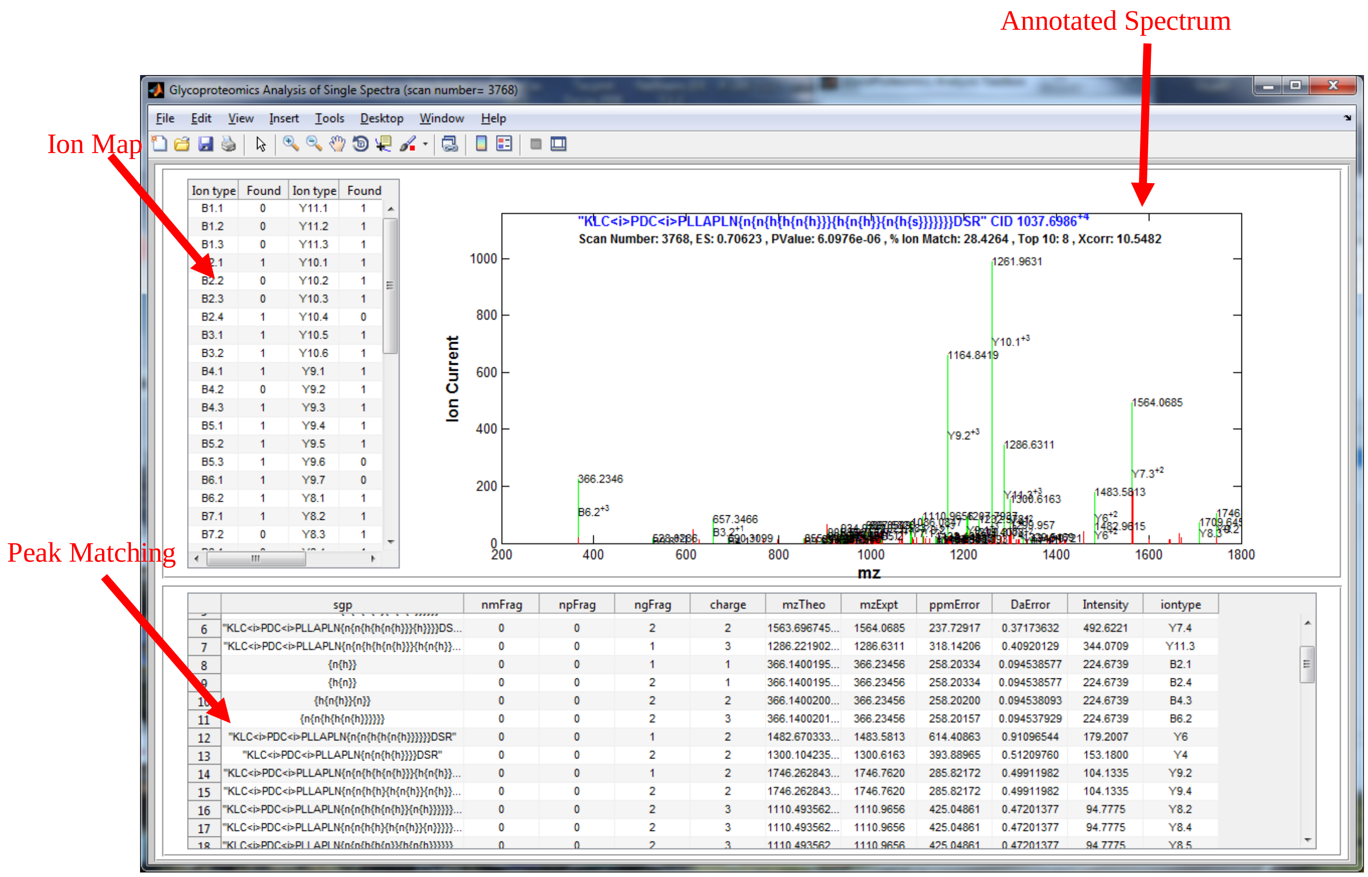
Once the score CSV file is loaded, the following information will be seen:

- 'Header' panel: This window displays the scoring parameters including the inputs used for 'Tandem MS Analysis'. This section also specifies the absolute location of the input data file. This file is needed for running this function.
- 'Scoring Table' panel: This table displays: 1) The FASTA header of the identified protein ('protein'); 2) The scan number of the MS^2 spectrum ('scan'); 3) Experimental mass of the precursor ion ('exptMass'); 4) Monoisotopic mass of the candidate glycopeptide ('monoMass'), 5) Most abundant mass of the candidate glycopeptide ('mostAbMass'); 6) Precursor ion charge state ('charge'); 7) identified glycopeptide in SmallGlyPep format ('peptide'); 8) Fragmentation mode ('fragMode'); 9) Xcorr htAvg parameter ('Xcorr'); 10) % of ions matched among all theoretical fragments ('percentIonMatch'); 11) *P*-value (pValue); 12) number of Top 10 peak matched ('top10') etc. Additional information is also provided in this table (not shown in above graphics) regarding fragmentation parameters, signature peaks identified, and more scoring parameters that are not currently used to calculate the Ensemble Score. Most significantly, the table provides the:
  * Ensemble Score ('ES'): This is a weighted average parameter of four different statistical metrics that score the candidate glycopeptide specified in the 'peptide' column.
  * Decoy Ensemble Score ('decoyES'): This parameter is the same as 'ES' only it is generated by creating a decoy of the identified candidate glycopeptide.
- 'FDR Option' panel: This section calculates the 'ES cut-off value' that is used to identify 'true positive' matches based on false discovery rate (FDR) calculations. Identifications below this cut-off value are removed from the display during this step. Three options are available for this FDR filtering: i. The % value for the FDR threshold; ii. Whether this % applies to specific fragmentation modes only; and iii. Whether this is applied to peptide, glycopeptide or both. Once these fields are specified using the pull-down menus, pressing the "Apply FDR filter" button results in ES cut-off calculations and the removal of peptide-spectrum-matches (PSMs) that are not 'true positive'. A button ('Reset to Original') is also available to undo the FDR filtering.
- 'Display Option': As an alternative to 'FDR Options', three additional filters are available below the FDR panel that enables simplification of the 'Scoring Table' display: 1. 'Show Glycan Only' removes all hits that are not glycopeptides; 2. 'Ensemble Score > value' retains all hits above a specified ES value; 3. A pull-down menu allows selection of display based on specified fragmentation modes.

Highlighting any row in the Scoring Table results in the display of the corresponding experimental MS^2 spectrum. In the case of an mzXML data file, a wait bar will appear in the first instance when the user selects a given row, since it takes time to load all spectra into the MATLAB memory space. Subsequent, row selection will be much more rapid in the order of seconds.


Loading mzXML file...Please wait...

An annotated spectrum and two associated tables ("**ion map**" and "**peak matching**") are presented for each chosen row. In this display (shown below) green color identifies peaks matched by fragments ions of the candidate glycopeptide, and red indicating peaks that could not be matched. For the matched peaks, its corresponding *mz* and ion type/charge are annotated in the figure. An "ion map" table is also shown in the left top panel. In the "ion map" table, all the possible theoretical ions are listed, with the number '1' indicating that a matching result was found in the experimental spectrum. In addition, a summary of other information related to scoring is displayed in the "**peak matching**" table. In this table, all the matched fragment ions are shown along with the *SmallGlyPep* string that corresponds to these ions. Additional information are also included to specify the fragmentation parameters, fragment ion type, theoretical mass, experimental mass etc.



"KLC<i>PDC<i>PLLAPLN{n{n{h{h{n{h}}}{h{n{h}}{n{h{s}}}}}}}DSR" CID 1037.6986^{+4}

Scan Number: 3768, ES: 0.70623 , PValue: 6.0976e-06 , % Ion Match: 28.4264 , Top 10: 8 , Xcorr: 10.5482

Ion type	Found	Ion type	Found
B1.1	0	Y11.1	1
B1.2	0	Y11.2	1
B1.3	0	Y11.3	1
B2.1	1	Y10.1	1
B2.2	0	Y10.2	1
B2.3	0	Y10.3	1
B2.4	1	Y10.4	0
B3.1	1	Y10.5	1
B3.2	1	Y10.6	1
B4.1	1	Y9.1	1
B4.2	0	Y9.2	1
B4.3	1	Y9.3	1
B5.1	1	Y9.4	1
B5.2	1	Y9.5	1
B5.3	1	Y9.6	0
B6.1	1	Y9.7	0
B6.2	1	Y8.1	1
B7.1	1	Y8.2	1
B7.2	0	Y8.3	1

	sgp	nmFrag	npFrag	ngFrag	charge	mzTheo	mzExpt	ppmError	DaError	Intensity	iontype
6	"KLC<i>PDC<i>PLLAPLN{n{n{h{h{n{h}}}{h}}}}DS...	0	0	2	2	1563.696745...	1564.0685	237.72917	0.37173632	492.6221	Y7.4
7	"KLC<i>PDC<i>PLLAPLN{n{n{h{h{n{h}}}{h{n{h}}...	0	0	1	3	1286.221902...	1286.6311	318.14206	0.40920129	344.0709	Y11.3
8	{n{h}}	0	0	1	1	366.1400195...	366.23456	258.20334	0.094538577	224.6739	B2.1
9	{h{n}}	0	0	2	1	366.1400195...	366.23456	258.20334	0.094538577	224.6739	B2.4
10	{h{n{h}}{n}}	0	0	2	2	366.1400200...	366.23456	258.20200	0.094538093	224.6739	B4.3
11	{n{n{h{h{n{h}}}}}}	0	0	2	3	366.1400201...	366.23456	258.20157	0.094537929	224.6739	B6.2
12	"KLC<i>PDC<i>PLLAPLN{n{n{h{h{n{h}}}}}}DSR"	0	0	1	2	1482.670333...	1483.5813	614.40863	0.91096544	179.2007	Y6
13	"KLC<i>PDC<i>PLLAPLN{n{n{h{h}}}}DSR"	0	0	2	2	1300.104235...	1300.6163	393.88965	0.51209760	153.1800	Y4
14	"KLC<i>PDC<i>PLLAPLN{n{n{h{h{n{h}}}{h{n{h}}...	0	0	1	2	1746.262843...	1746.7620	285.82172	0.49911982	104.1335	Y9.2
15	"KLC<i>PDC<i>PLLAPLN{n{n{h{h}{h{n{h}}{n{h}}...	0	0	2	2	1746.262843...	1746.7620	285.82172	0.49911982	104.1335	Y9.4
16	"KLC<i>PDC<i>PLLAPLN{n{n{h{h{n{h}}{n{h}}}}}}...	0	0	2	3	1110.493562...	1110.9656	425.04861	0.47201377	94.7775	Y8.2
17	"KLC<i>PDC<i>PLLAPLN{n{n{h{h}{h{n{h}}{n}}}}}...	0	0	2	3	1110.493562...	1110.9656	425.04861	0.47201377	94.7775	Y8.4
18	"KLC<i>PDC<i>PLLAPLN{n{n{h{h{n}}{h{n{h}}}}}}	0	0	2	3	1110.493562	1110.9656	425.04861	0.47201377	94.7775	Y8.5

Note: A beta-version of a new spectrum annotation software is currently being tested. The focus of this program is on: i. Improving presentation of final results using the newly released SNFG (Symbolic nomenclature for glycan) carbohydrate representation and our software program DrawGlycan-SNFG (www.VirtualGlycome.org/DrawGlycan). ii. Comparing the scores of isomeric glycopeptides, in order to distinguish between closely related carbohydrates. This version will be released once program development has been completed.

6. Using MATLAB scripts for GlycoProteomics Analysis: *for MATLAB user/developer willing to write code*

This section provides examples for the basic usages of GlycoPAT functions in the MATLAB command window. Detailed usage examples are provided with the help documentation (see Chapter 4 for instructions). All data files associated used in the examples below are stored in <`GlycoPATInstallationDIR>`/toolbox/test.

The functions of GlycoPAT can be categorized into five groups, viz. **Mass spectrometry data handling**, **Glycopeptide mass calculation, Glycopeptide digestion, Glycopeptide fragmentation** and **Spectrum scoring**.

6.1 Mass Spectrometry Handling.

The commands in this category can be divided into three subgroups:

- **MS file input**: This subgroup contains two commands: **readmzXML** and **readmzDTA**, The function and usage of each command are as follows:

  a) **readmzXML** reads MS data from an mzXML file. Below is one example:

  ❖**Example 1**: **readmzXML** reads mass spectrometry data from a local file (`'fetuin_test.mzxml'`) and returns an object (`Mzxml_data`) of MATLAB mzXML class.

```
Mzxml_data = readmzXML('fetuin_test.mzXML');  % The command assumes that
                                                this mzXML file is in the
                                                MATLAB search path
```

  b) **mzDTAread** reads MS^2 data from a DTA file directory.

  ❖ **Example 2**: **mzDTAread** reads all MS^2 spectra from a file directory where the DTA files are stored (`e.g., c:\glycopat\toolbox\test\data\dta`), and returns a MATLAB structure `dta_data` that contains four field: scan, MH, z and spectra.

```
dta_data = readmzDTA('c:\glycopat\toolbox\test\data\dta\fetuin');
```

- **MS spectra summary and retrieval:** Three commands (**retrieveSpectraSummary**, **retrieveMSSpectra**, **retrieveActMethod**) are provided to retrieve spectra-related information from an mzXML object.

  a) **retrieveSpectraSummary** retrieves the number of MS^1 and MS^2 spectra, and the number of spectra in different fragmentation modes. An example is shown below:

  ❖**Example 3: reads an mzXML object and returns the summary of MS^1/MS^2 spectra.**

```
mzXMLobj  = readmzXML('fetuin_test.mzXML'); % The command
summary   = mzXMLobj.retrieveSpectraSummary;   %assumes that this
disp(summary)                                   % mzXML file is in
                                             % the MATLAB search path
```

```
Answer:
     Numscan    :  401
     numscanms1:  57
     numscanms2:  344
     numetdms2:   172
     numcidms2:   172
     numhcdms2:   0
```

b) **retrieveMSSpectra** retrieves MS spectrum from an mzXML object. Below is an example.

❖**Example 4: reads an mzXML object and returns the spectrum based on the scan number specified by the user, in this case scan 34.**

```
        mzXMLobj  = readmzXML('fetuin_test.mzXML');
        spectra_3500thscan= mzXMLobj.retrieveMSSpectra('scannum',3500);
        whos spectra_3500thscan
  Answer:
      Name                  Size            Bytes  Class     Attributes

    spectra_3500thscan      738x2             5904  single
```

c) **retrieveActMethod** retrieves spectra activation method from an mzXML object. An example is given as below.

❖**Example 5: reads an mzXML object and returns the spectra activation method for scan number 3500.**

```
        mzXMLobj    = readmzXML('fetuin_test.mzXML');
        actmethod_3500thscan   = mzXMLobj.retrieveActMethod(...
        'scannum',3500);
        disp(actmethod_3500thscan)
  Answer:
     CID
```

6.2 Glycopeptide Mass Calculation

Eight functions are available in **GlycoPAT** to compute glycopeptide formula and mass. They are categorized into the following two groups.

▪**Formula calculation.**

This group deals with formula calculation. This includes four functions: '**ptmformula**', '**pepformula**', '**glypepformula**', '**glyformula**'. An example for "glypepformula' is shown below:

❖**Example 6: glypepformula reads the glycopeptide in "SmallGlyPep" format and computes its elemental formula**

```
glypepformulaObj =glypepformula('T{n{h{s}}{n{f}{h{s}}}}EPPRAM<o>M<o>D')
    disp(glypepformulaObj.cfstruct)

    Answer:
        cfstruct: [1x1 struct]
           C: 98
           H: 160
           N: 16
           O: 57
           S: 2
```

▪**Mass calculation**

The group contains four functions that compute the molecular weight or mass to charge ratio. They are **ptm, pepmw, glymw, glypepmw.** An example for **glypepmw** is below:

❖**Example 7: glypepmw reads the glycopeptide in "SmallGlyPep" format and compute its mass or m/z ratio.**

For example to calculate the y-ion with the following SGP formula at z=2:

```
 finalMW=glypepMW('FLPET{n{h}}EPPRPM<o>M<o>D','y',2)
```

```
Answer:
    finalMW =
             978.926833785
```

To calculate the SGP formula weight of full glycopeptide at z=1:

```
finalMW=glypepMW('FLPET{n{h}}EPPRPM<o>M<o>D','full',1)
Answer:
    finalMW =
             1955.8380175
```

6.3 Glycopeptide Digestion

Eight functions are provided in this category. These functions include digestion option setup functions (**fixedptmset**, **varptmset**, **digestoptionset),** digestion file input functions (**peptideread**, and **fixedptmread**, **varptmread)** and SGP digestion functions (**digestSGP, cleaveProt)**.

▪**Digestion File Input**

This group contains three functions. They can be used to read protein sequence, fixed PTM modification and variable PTM modification (including glycan and non-glycan type) from the local file, respectively.

❖**Example 8: peptideread retrieves protein sequence information from a FASTA file.**

```
pepseq = peptideread('19Fc.txt');
disp(pepseq)
Answer:
 Header:'19Fc'
 Sequence: [1×261 char]
```

▪**Option setup**

This group contains **fixedptmset**, **varptmset** and **digestoptionset** functions**. fixedptmset** sets the fixed posttranslational modification, **varptmset** sets the variable modification, and **digestoptionset** set the options for peptide digestion. An example of each of these functions is shown below:

❖**Example 9: fixedptmset** sets a mass modification of `57.0214617` to all cysteine residue.

```
fptmopt = fixedptmset('aaresidue','c','mod','i','mw',57.0214617);
```

❖**Example 10: varptmset** sets up either oxidation of Met (first line below), or two potential glycan modifications at the 18th amino acid (second and third line below).

```
vptmopt = varptmset('aaresidue','M','mod','o','pos','0');
vptmopt = varptmset(vptmopt,'aaresidue','X','mod','{n{h{s}}} ',...
'pos','18');
vptmopt = varptmset(vptmopt,'aaresidue','X','mod','{n{h}s}',...
'pos','18');
disp(vptmopt)

Answer:
   aaresidue: {3x1 cell}
         mod: {3x1 cell}
         pos: {3x1 cell}
```

❖**Example 11: digestoptionset** sets all digestion options including specification of the fixed and variable PTM modifications. Example:

```
fptmopt = fixedptmset('aaresidue','c','mod','i','mw',57.0214617);
vptmopt = varptmset('aaresidue','M','mod','o','pos','0');
vptmopt = varptmset(vptmopt,'aaresidue','X','mod','{n{h{s{}}}}
    ','pos','18');
options = digestoptionset('missedmax',1,'minpeplen',3,'maxptm',2);
options = digestoptionset(options,'fixedptm',fptmopt,...
   'varptm',vptmopt);
disp(options)

Answer:
   options =
         missedmax: 1
         minpeplen: 3
            maxptm:  2
         maxpeplen: 30
            minptm:  0
      isoutputfile: 'no'
    outputfilename: 'digestedpep.txt'
            varptm: [1x1 struct]
          fixedptm: [1x1 struct]
```

▪**Protein Digestion**

This group contains the main digestion function (**digestSGP)** and internal function **(cleaveProd). digestSGP** is used to cleave protein with glycan modification.

❖**Example 12: digestSGP** creates a list of enzyme-digested fragments given the protein sequence, list of glycan and non-glycan modifications and other digestion parameters (such as minimum and maximum peptide length, minimum and maximum number of modifications, the number of missing cleavages).

```
options = digestoptionset('missedmax',1,'minpeplen',3,'maxptm',2);
varptm  = varptmread('variableptm.txt');
fixedptm  = fixedptmread('fixedptm.txt');
options=digestoptionset(options,'fixedptm',fixedptm,'varptm',varptm)
glycopepfrag = digestSGP('19Fc.txt','Gluc',options);
disp(glycopepfrag);

Answer:
'ALHNHYTQKSLSLSPGKHHHHHH'
     'DPE'
     'DPEVKFNWYVDGVE'
     'EQYNSTYRVVSVLTVLHQDWLNGKE'
     'EQYN{n{f}{n{h{h{n{h{s}}}}{h{n{h{s}}}}}}}STYRVVSVLTVLHQDWLNGKE'
'EQYN{n{f}{n{h{h{n{h}}{n{h}}}{h{n{h}}{n{h}}}}}}STYRVVSVLTVLHQDWLNGK
E'
     'EQYN{n{f}{n{h{h{n{h}}}{h{n{f}{h{s}}}}}}}STYRVVSVLTVLHQDWLNGKE'
     'EQYN{n{f}{n{h{h{n{h}}}{h{n{h{n}}}}}}}STYRVVSVLTVLHQDWLNGKE'
     'EQYN{n{f}{n{h{h{n{h}}}{h{n{h{s}}}}}}}STYRVVSVLTVLHQDWLNGKE'
     'EQYN{n{f}{n{h{h{n{h}}}{h{n{h}}}}}}STYRVVSVLTVLHQDWLNGKE'
     'EQYN{n{f}{n{h{h{n}}{h{n{h{n}}}}}}}STYRVVSVLTVLHQDWLNGKE'
     'EQYN{n{f}{n{h{h{n}}{h{n{h}}}}}}STYRVVSVLTVLHQDWLNGKE'
     'EQYN{n{f}{n{h{h{n}}{h{n}}}}}STYRVVSVLTVLHQDWLNGKE'
     'KTISKAKGQPRE'
     'KTISKAKGQPREPQVYTLPPSRDE'
     'LAGAPSVFLFPPKPKDTLM<o>ISRTPE'
     'LAGAPSVFLFPPKPKDTLMISRTPE'
     'LTKNQVSLTC<i>LVKGFYPSDIAVE'
```

```
'LTKNQVSLTC<i>LVKGFYPSDIAVEWE'
'PPRPM<o>M<o>DDDDKSRTC<i>PPC<i>PAPE'
'PPRPM<o>MDDDDKSRTC<i>PPC<i>PAPE'
'PPRPMM<o>DDDDKSRTC<i>PPC<i>PAPE'
'PPRPMMDDDDKSRTC<i>PPC<i>PAPE'
'PQVYTLPPSRDE'
'QYNSTYRVVSVLTVLHQDWLNGKE'
'QYN{n{f}{n{h{h{n{h{s}}}}{h{n{h{s}}}}}}}STYRVVSVLTVLHQDWLNGKE'
…. and so on
```

6.4 Glycopeptide Fragmentation

This category provides six commands, viz., **breakGlyPep**, **joinGlyPep**, **glycanFrag**, **multiSGPFrag, UQFragIon** and **compileFrags**. The first two functions**, breakGlyPep** and **joinGlyPep** can be used to assemble the glycopeptide or break it into the peptide, glycan and other modification components. The other functions **glycanFrag** and **multiSGPFrag** are designed to perform the fragmentation of the glycopeptide. The commands **UQFragIon** and **compileFrags** remove duplicate fragment ions and consolidate all fragment ions into a list.

Below we show two examples of **multiSGPFrag**. The usage of other commands can be found using "help functionname".

❖**Example 13: multiSGPFrag** reads the sequence of a glycopeptide, fragmentation parameters (# of glycan, peptide, and modification fragment points), and the charge state, and returns all fragment ions. Here, since ngFrag=2, up to two glycan fragmentations are allowed in the product ion.

```
SmallGlyPep='VPT{n{h}}T{n{h{s}}}AASTPDAVDK';
nmFrag=0;
npFrag=0;
ngFrag=2;
z=1;
AllFragIons=multiSGPFrag(SmallGlyPep,nmFrag,npFrag,ngFrag,z)
Answer:
       AllFragIons = 1x18 struct array with fields:
                     original
                     sgp
                     nmFrag
                     npFrag
                     ngFrag
                     mz
                     type
                     charge
```

To look at any specific ion, say the 15th one, type:

```
AllFragIons(15)
Ans=
    original: 'VPT{n{h}}T{n{h{s}}}AASTPDAVDK'
         sgp: '{n{h{s}}}'
      nmFrag: 0
      npFrag: 0
      ngFrag: 1
          mz: 657.235435932
        type: '-b{n{h{s}}}'
      charge: 1
```

❖**Example 14: multiSGPFrag** reads the sequence of a glycopeptide, fragmentation parameters (# of glycans, peptide, and modification fragment points), and the charge state. It returns all fragments produced by cleavage of the peptide backbone at a single site (ngFrag=0,nmFrag=0,npFrag=1):

```
 SmallGlyPep='VPT{n{h}}T{n{h{s}}}AASTPDAVDK';
nmFrag=0;
npFrag=1;
ngFrag=0;
z=1;
AllFragIons=multiSGPFrag(SmallGlyPep,nmFrag,npFrag,ngFrag,z)
Answer:
          AllFragIons = 1x53 struct array with fields:
                              original
                              sgp
                              nmFrag
                              npFrag
                              ngFrag
                              mz
                              type
                              charge
```

To view the 15th ion, type:

```
AllFragIons(15)
 ans =
     original: 'VPT{n{h}}T{n{h{s}}}AASTPDAVDK'
           sgp: 'V'
       nmFrag: 0
       npFrag: 1
       ngFrag: 0
            mz: 117.102787836
          type: 'c1'
       charge: 1
```

6.5 Spectra Scoring

Ten functions are available to facilitate MS^2 spectra scoring. Here experimental MS^2 are compared against a library of theoretical glycopeptides generated using the **digestSGP** function. These include:

- **scoreCSVwrite/scoreCSVread** reads/writes scoring results (ensemble scores, glycopeptide hits, spectra scan number and many other parameters) to a local csv file
- **spectracmp** finds the matches between experimental spectrum and the theoretical spectrum.
- **scoreProb** calculates the *P*-value for the theoretical match.
- **CrossCorr** calculates the cross correlation value for the experimental spectrum.
- **swapAcid** generates the decoy for the target glycopeptide by changing the peptide sequence.
- **fdrdecoy** generates the decoy for the target glycopeptide by changing the peptide sequence and the mass for every monosaccharides in the glycan.
- **score1Spectrum/scoreAllSpectra** computes the ensemble score for either one single or for all MS^2 spectra.
- **fdr** calculates the false discovery rate based on decoy datasets

- **fdrfilter** filters the PSM using false discovery rate
- **fdrdecoy** generates decoy glycopeptide for FDR calculation
- **SignatureGlycan** finds signature peaks in the spectrum.

Below we show one example for **fdrdocy** and **scoreAllSpectra**. The usages and examples of other functions can be seen using “help functionname”.

Example 15: **fdrdecoy** generates the decoy glycopeptide for fdr computation

```
SmallGlyPep='M<o>{n{h{s}}}GHKL<o>FLM{n{h{s}}}L';
type='flip';
Decoy=fdrdecoy(SmallGlyPep, type)

Answer:
LM{241.9748{147.3541{266.8986}}}LFL<o>KHGM<o>{183.8411{160.0191{312.3674}
}} % different numerical values will appear since there is a random number generator
```

Example 16: **scoreAllSpectra** scores all the spectra stored in ‘fetuin_test.mzXML’file using the glycopeptide library in 'digestedfetuin_Nglycanonly.txt ' obtained from the digestSGP command. The mzXML file is stored in <GlycoPAT InstallationDirectory>\toolbox\test\digest. The output including the scoring parameter are saved in a local file named, testscore.csv.

```
glycopatroot = 'c:\glycopat\'; % replace “c:\glycopat’ with GlycoPAT installation directory
pepfile= 'digestedfetuin_Nglycanonly.txt ';
xmlfilepath = fullfile(glycopatroot, 'toolbox', 'test', 'data', 'mzxml');
pepfilefullname = fullfile(xmlfilepath,pepfile);
mzxmlfilename = 'fetuin_test.mzXML';
fragMode = 'AUTO';
MS1tol = 10.000000;
MS1tolUnit = 'ppm';
MS2tol = 1.000000;
MS2tolUnit = 'Da';
OutputDir = xmlfilepath;
OutCSVname = 'testscore_cid.csv';
maxlag = 50;
CutOffMed = 2.000000;
FracMax = 0.020000;
nmFrag = 0;
npFrag = 2;
ngFrag = 0;
selectPeak =[163.1,292.1,366,454.1,657.2];
calchit = scoreAllSpectra(pepfilefullname,xmlfilepath,fragMode,MS1tol,MS1tolUnit,MS2tol, ...
MS2tolUnit,OutputDir,OutCSVname,maxlag,CutOffMed,FracMax,nmFrag,npFrag,ngFrag, ...
selectPeak,false,mzxmlfilename);

Answer:
    k = 1
    calculating isotopic mass
i =
       1
scan number: 3400
```

```
possible glycopeptide:
SmallGlyPep =
        []
i =
        2
scan number: 3401
possible glycopeptide:
SmallGlyPep =
        []
i =
        3
…. (scores 340 files and saves results in 'testscore_cid.csv').
```

6.6 Custom monosaccharides and PTM modifications

The object-oriented, modular design of GlycoPAT allows user-defined extensions. Below are two examples about how to extend the code for the definition of new types of monosaccharides and protein PTM modifications.

- **Definition of custom monosaccharides:** The current monosaccharide database has 10 members that are stored in the "Glycan" class (located in ~toolbox\class\@Glycan\). This class contains different elements:
    - Full glycan name: `glyfullname = {'Hex','HexNAc',...`
    - Single letter name: `gly1let     = {'h','n',`
    - Glycan formula:
      ```
      glyformula = {struct('C',6,'H',10,'O',5),...
      struct('C',8,'H',13,'O',5,'N',1),..
      ```
    - Container map describing the relation between single letter monosaccharide name and glycan chemical formula:
      ```
      glycanformulaMap = containers.Map({'h','n'...
                            ...
                            ...{struct('C',6,'H',10,'O',5),...
                            struct('C',8,'H',13,'O',5,'N',1),...
      ```
    - Container map describing the relation between single letter monosaccharide name and molecular mass:
      ```
      glycanMSMap = containers.Map({'h','n'...
                     ...
                     ...{162.0528235,203.0793724,...
      ```

  Modify these variables in the "Glycan" class to add new members by appending new full names, single letter names, formula, and by similarly adding corresponding elements to the Container.Map elements.

- **Definition of a new type of protein PTM modifications**: Similar to above, new PTM modifications can be added by editing the Modification.m file located at ~toolbox\class\@Modification\. Here, elements need to be appended to the following properties:
    - Single letter: `mod1let     = {'o','s',...`
    - Full name of modification: `modfullname = {'oxidation','sulfation',...`
    - In *modadd*, write '1' if molecular mass increases upon modification and '0' if it decreases: `modadd      = {1,1,...`
    - Modification formula:
      ```
      modformula  = {struct('O',1),...
                     struct('S',1,'O',3),...
      ```
    - Container.Map with PTM formula: `formulaMap=Containers.Map({'o','s'...`

```
                                                      ...
                                                      ... {struct('O',1),...
                                                       struct('S',1,'O',3),...
```

- ○ Container.Map containing modadd data indicating either an increase or decrease in molecular mass: `mwEffectMap = containers.Map({'o','s',...`

```
                                                       ...  {1,1,...
```

Note that there is some duplication of data in the Class definitions above since the goal is to store both constants that can be used for later developments and Container.Map objects that are used in the current version of GlycoPAT.

6.7 Glycan library generation

Custom 'glycan search databases' can be created in GlycoPAT using functions that were originally developed for a software GNAT: Glycosylation Network Analysis Toolbox (Liu, Puri et al. 2013; Liu and Neelamegham 2014). This is an open-source software available from sourceforge (manual stored in ~toolbox/matlablib/gnatmanual.pdf). Functions related to this program are included with the GlycoPAT package.

In particular, GlycoPAT uses the *connection inference algorithm* ("*inferGlyConnPath*") of GNAT to build the glycan search library (Liu and Neelamegham 2014). This algorithm connects a list of input glycans (available from existing literature or Glycomics studies) using enzyme-substrate specificity definitions that are provided in the enzyme (*Enz*) class of GNAT. Thus, the two inputs needed for *inferGlyConnPath* are:

- An array of glycans: These can be written in XML format using GlycoWorkbench (https://code.google.com/p/glycoworkbench/) or other software that can write glycoct_xml format inputs. Please refer to the GNAT manual for detailed step by step guidelines or follow the example below. The example below reads two glycans in XML format, Man9.glycoct_xml and tri_SSS.glycoct_xml.
- An array of enzymes: The enzymes which include both glycosyltransferases (*GTEnz*) and glycosidases (*GHEnz*). They participate in the biosynthetic pathways that link the glycans defined in the previous step. For the specific example provided below, several such examples of GTEnz and GHEnz are defined *in silico* including sialyltransferases (siaT), fucosyltransferases (fucT8), GlcNAcTransferases (mgat 1,2,4,5), manosidases (manI and its variants, manII) and galactosyltransferases (beta4galt).

Following the above definition, the *inferGlyConnPath* program is run to generate the biosynthetic pathway that links the 'array of glycans' using the 'array of enzymes'. In the final step, the *clusterglycans* function is applied to convert the glycan structures generated by the program into SmallGlyPep (SGP1.0) nomenclature that can be copy-pasted into the variableptm input file of GlycoPAT.

For the user's convenience, the pathway generated using *inferGlyConnPath* is also displayed using the GNV: Glycosylation Network Visualizer.

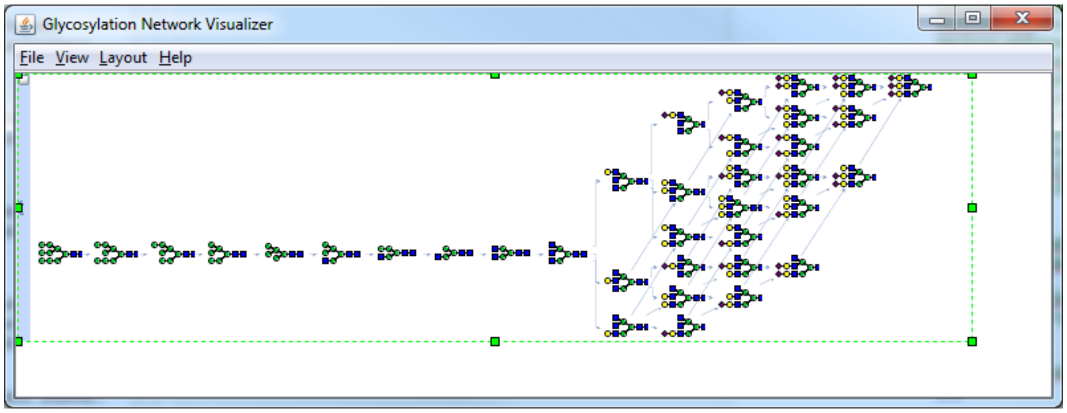
```
clc; clear;
substrateArray = CellArrayList;
s1species = GlycanSpecies(glycanMLread('Man9.glycoct_xml'));
s2species = GlycanSpecies(glycanMLread('tri_SSS.glycoct_xml'));
```

```
substrateArray.add(s1species);
substrateArray.add(s2species);

residueMap       = load('residueTypes.mat');
manResType       = residueMap.allresidues('Man');
m3gn             = glycanMLread('m3gn.glycoct_xml');

%% define sia T
siaT                   = GTEnz([2;4;99;6]);
siaT.isTerminalTarget = true;
siaT.resfuncgroup      = residueMap.allresidues('NeuAc');
siaTbond               = GlycanBond('3','2');
siaT.linkFG            = struct('anomer','a','bond',siaTbond);
galResType             = residueMap.allresidues('Gal');
galBond                = GlycanBond('4','1');
siaT.resAtt2FG         = galResType;
siaT.linkresAtt2FG     = struct('bond', galBond,'anomer','b');
siaT.substNABranch     = glycanMLread('NGlycanBisectGlcNAc.glycoct_xml');
siaT.targetbranchcontain = glycanMLread('gntetargetbranch.glycoct_xml');

%% Define Fuc T
fucT8                      = GTEnz([2;4;1;68]);
fucT8.isTerminalTarget     = false;
fucT8.resfuncgroup         = residueMap.allresidues('Fuc');
fuctbond                   = GlycanBond('6','1');
fucT8.linkFG               = struct('anomer','a','bond',fuctbond);
glcnacResType              = residueMap.allresidues('GlcNAc');
glcnacBond                 = GlycanBond('?','?');
fucT8.resAtt2FG            = glcnacResType;
fucT8.linkresAtt2FG        = struct('bond', glcnacBond,'anomer','?');
fucT8.targetNABranch       = glycanMLread('NGlycanBisectGlcNAc.glycoct_xml');
fucT8.substNAResidue       = residueMap.allresidues('Gal');
fucT8.substMinStruct       = m3gn;

%% Define MGAT1
mgat1                      = GTEnz([2;4;1;101]);
mgat1.resfuncgroup         = residueMap.allresidues('GlcNAc');
glcnacbond                 = GlycanBond('2','1');
mgat1.linkFG               = struct('anomer','b','bond',glcnacbond);
manBond                    = GlycanBond('3','1');
mgat1.resAtt2FG            = manResType;
mgat1.linkresAtt2FG        = struct('bond', manBond,'anomer','a');
mgat1.substMinStruct       = glycanMLread('M5.glycoct_xml');
mgat1.substMaxStruct       = glycanMLread('M5.glycoct_xml');

%% Define MGAT2
mgat2                      = GTEnz([2;4;1;143]);
residueMap                 = load('residueTypes.mat');
mgat2.resfuncgroup         = residueMap.allresidues('GlcNAc');
glcnacbond                 = GlycanBond('2','1');
mgat2.linkFG               = struct('anomer','b','bond',glcnacbond);
manResType                 = residueMap.allresidues('Man');
manBond                    = GlycanBond('6','1');
mgat2.resAtt2FG            = manResType;
mgat2.linkresAtt2FG        = struct('bond', manBond,'anomer','a');
mgat2.isTerminalTarget     = true;
mgat2.substMinStruct       = m3gn;
mgat2.targetBranch         = glycanMLread('mgat2actingbranch.glycoct_xml');
mgat2.substNABranch        = CellArrayList;
mgat2.substNABranch.add(glycanMLread('NGlycanBisectGlcNAc.glycoct_xml'));
mgat2.substNABranch.add(glycanMLread('mgat2substrateNAbranch.glycoct_xml'));

%% Define MANI
```

```
man1b1                       = GHEnz([3;2;1;113]);
man1b1.resfuncgroup          = manResType;
man1b1.linkFG.anomer         ='a';
manBond                      = GlycanBond('2','1');
man1b1.linkFG.bond           = manBond;
man1b1.resAtt2FG             = manResType;
man1b1AttachBond             = GlycanBond('3','1');
man1b1.linkresAtt2FG         = struct('bond', man1b1AttachBond,'anomer','a');
man1b1.substMaxStruct        = glycanMLread('M9.glycoct_xml');
man1b1.substMinStruct        = glycanMLread('M9.glycoct_xml');
man1b1.substNAResidue        = residueMap.allresidues('Gal');

man1a1                       = GHEnz([3;2;1;113]);
man1a1.resfuncgroup          = manResType;
man1a1.linkFG.anomer         ='a';
manBond                      = GlycanBond('2','1');
man1a1.linkFG.bond           = manBond;
man1a1.resAtt2FG             = manResType;
man1a1AttachBond             = GlycanBond('2','1');
man1a1.linkresAtt2FG         = struct('bond', man1a1AttachBond,'anomer','a');
man1a1.substMinStruct        = glycanMLread('M8K.glycoct_xml');
man1a1.substMaxStruct        = glycanMLread('M8K.glycoct_xml');
man1a1.substNAResidue        = residueMap.allresidues('Gal');

man1a2                        = GHEnz([3;2;1;113]);
man1a2.resfuncgroup           = manResType;
man1a2.linkFG.anomer          ='a';
manBond                       = GlycanBond('2','1');
man1a2.linkFG.bond            = manBond;
man1a2.resAtt2FG              = manResType;
man1a2AttachBond              = GlycanBond('6','1');
man1a2.linkresAtt2FG          = struct('bond', man1a2AttachBond,'anomer','a');
man1a2.substMinStruct         = glycanMLread('M7K.glycoct_xml');
man1a2.substMaxStruct         = glycanMLread('M7K.glycoct_xml');
man1a2.substNAResidue         = residueMap.allresidues('Gal');

man1c1                        = GHEnz([3;2;1;113]);
man1c1.resfuncgroup           = manResType;
man1c1.linkFG.anomer          ='a';
manBond                       = GlycanBond('2','1');
man1c1.linkFG.bond            = manBond;
man1c1.resAtt2FG              = manResType;
man1c1AttachBond              = GlycanBond('3','1');
man1c1.linkresAtt2FG          = struct('bond', man1c1AttachBond,'anomer','a');
man1c1.substMaxStruct         = glycanMLread('M6K.glycoct_xml');
man1c1.substMinStruct         = glycanMLread('M6K.glycoct_xml');
man1c1.substNAResidue         = residueMap.allresidues('Gal');

%% Define MANII
manResType                    = residueMap.allresidues('Man');
manii                         = GHEnz([3;2;1;114]);
manii.resfuncgroup            = manResType;
manBond(1,1)                  = GlycanBond('3','1');
manBond(2,1)                  = GlycanBond('6','1');
manii.linkFG                  = struct('bond',manBond ,'anomer','a');
manii.resAtt2FG               = manResType;
resbond                       = GlycanBond('6','1');
manii.linkresAtt2FG           = struct('bond', resbond,'anomer','a');
manii.substNABranch           = glycanMLread('1008.51.glycoct_xml');
manii.substMaxStruct          = glycanMLread('1824.91.glycoct_xml');

%% Definition and visualization of MGAT4 enzyme
mgat4                          = GTEnz([2;4;1;145]);
```

```
mgat4.resfuncgroup              = residueMap.allresidues('GlcNAc');
manResType                     = residueMap.allresidues('Man');
manBond                         = GlycanBond('3','1');
mgat4.resAtt2FG                = manResType;
mgat4.linkresAtt2FG             = struct('bond', manBond,'anomer','a');
glcnacbond                      = GlycanBond('4','1');
mgat4.linkFG                    = struct('anomer','b','bond',glcnacbond);
mgat4.substMinStruct            =
glycanMLread('mgat4_5substMinStruct_K.glycoct_xml');
mgat4.substNABranch             = CellArrayList;
mgat4.substNABranch.add(glycanMLread('NGlycanBisectGlcNAc.glycoct_xml'));
mgat4.substNABranch.add(glycanMLread('mgat4subsNAbranch.glycoct_xml'));
mgat4.substNAResidue            = residueMap.allresidues('Gal');

%% Definition and visualization of MGAT5 enzyme
mgat5                           = GTEnz([2;4;1;155]);
mgat5.resfuncgroup              = residueMap.allresidues('GlcNAc');
manResType                      = residueMap.allresidues('Man');
manBond                         = GlycanBond('6','1');
mgat5.resAtt2FG                 = manResType;
mgat5.linkresAtt2FG             = struct('bond', manBond,'anomer','a');
glcnacbond                      = GlycanBond('6','1');
mgat5.linkFG                    = struct('anomer','b','bond',glcnacbond);
mgat5.substMinStruct            =
glycanMLread('mgat4_5substMinStruct_K.glycoct_xml');
mgat5.substNABranch             = CellArrayList;
mgat5.substNABranch.add(glycanMLread('NGlycanBisectGlcNAc.glycoct_xml'));
mgat5.substNABranch.add(glycanMLread('mgat5substrateNAbranch.glycoct_xml'));
mgat5.substNAResidue            = residueMap.allresidues('Gal');

beta4galt                        = GTEnz([2;4;1;38]);
beta4galt.isTerminalTarget        = true;
beta4galt.resfuncgroup            = residueMap.allresidues('Gal');
glcnacResType                    = residueMap.allresidues('GlcNAc');
glcnacBond                        = GlycanBond('?','1');
beta4galt.resAtt2FG              = glcnacResType;
beta4galt.linkresAtt2FG            = struct('bond', glcnacBond,'anomer','b');
galtbond                           = GlycanBond('4','1');
beta4galt.linkFG                    = struct('anomer','b','bond',galtbond);

enzArray = CellArrayList;
enzArray.add(man1b1);
enzArray.add(man1a1);
enzArray.add(man1a2);
enzArray.add(man1c1);
enzArray.add(manii);
enzArray.add(mgat1);
enzArray.add(mgat2);
enzArray.add(mgat4);
enzArray.add(mgat5);
enzArray.add(beta4galt);
enzArray.add(siaT);
enzArray.add(fucT8);


[isPath, nglycanpath]=inferGlyConnPath(substrateArray, enzArray);
nglycanpath2 = removeLinkageIsomerStruct(nglycanpath);
if(isPath)
   glycanPathViewer(nglycanpath2);
   fprintf(1,'After removing the linkage isomer, the final network has: \n');
   fprintf(1,'    %i  reactions\n',nglycanpath2.getNReactions);
   fprintf(1,'    %i  species\n',   nglycanpath2.getNSpecies);
   glycluster = clusterglycans(nglycanpath2);
```

```
 end

Output:
the round: 1
the number of total species in the pathway: 37
the number of total reactions in the pathway: 65
After removing the linkage isomer, the final network has:
    63  reactions
    36  species
```



7. Category List of Functions/Classes

GUIs

glycopatgui	- Open Menu Selection GUI
digestgui	- Open Digestion GUI
scoregui	- Open Scoring GUI
browsegui	- Open Browse Result GUI
spectraAnnotationgui	- Open Single Spectrum Annotation GUI
fraggui	- Open Fragmentation GUI

Mass Spectrometry Data Handling

readmzXML	- Read mzXML file
readmzDTA	- Read .dta file
retrieveSpectraSummary	- mzXML spectra summary
retrieveMSSpectra	- Retrieve spectra intensity list
retrieveActMethod	- Retrieve spectra activation method
PolishSpectra	- Remove noise from spectra globally
PolishSpectraLocal	- Remove local noise from spectra
removePrecursorIon	- Remove precursor ion peaks from spectra

Digestion

peptideread	- Read peptide sequence file
fixedptmread	- Read fixed PTM file
varptmread	- Read variable PTM file
varptmset	- Set variable posttranslational modification
fixedptmset	- Set fixed posttranslational modification
cleaveProt	- Digest single protein with protease regardless of PTM
digestoptionset	- Set digestion option
digestSGP	- Glycopeptide digestion

Fragmentation

compileFrags	- Consolidation of fragment ions
UQFragIon	- Find unique ions in fragment ion list
joinGlyPep	- Construct a glycopeptide in SmallGlyPep
breakGlyPep	- Break glycopeptide into peptide, glycan and other PTMs
multiSGPFrag	- Glycopeptide fragmentation

Glycopeptide mass

ptmformula	- Compute chemical formula of PTM
pepformula	- Compute chemical formula of peptide
glyformula	- Compute chemical formula of glycan

• **glypepformula**	- Compute chemical formula of glycopeptide
• **ptm**	- Calculate PTM monoisotopic mass
• **glyMW**	- Calculate glycan monoisotopic mass and m/z
• **pepMW**	- Calculate peptide monoisotopic mass and m/z
• **glypepMW**	- Calculate glycopeptide monoisotopic mass and m/z

Decoy

• **swapAacid**	- Swap amino acid sequence
• **glycanDecoy**	- Generate decoy glycan
• **fdrdecoy**	- Generate decoy (glycol)peptide
• **fdr**	**-** Calculate false discovery rate
• **fdrfilter**	**-** Filter the peptide/glycopeptide spectrum match using FDR

Scoring

• **spectracmp**	- Compare spectra
• **SignatureGlycan**	- Find signature peak in a spectrum
• **CrossCorr**	- Calculate cross-correlation parameter
• **scoreProb**	- Calculate multiple scoring parameters
• **score1Spectrum**	- Score one MS2 spectrum against the glycopeptide
• **scoreAllSpectra**	- Score all MS2 spectra
• **scoreCSVread**	- Read score csv file
• **scoreCSVwrite**	- Write score csv file

High performance computing

• **scoreAllSpectra_parfor**	- score using task-based parallel computing method
• **scoreAllSpectra_spmd**	- score using data-distributed parallel computing method

Note: These functions will work on a normal desktop and will use all available workers (i.e. CPU cores), provided the MATLAB parallel computing toolbox is installed. This is advantageous when handling large data files.

Both functions have also been tested on high performance computing clusters using the MATLAB parallel computing toolbox (www.mathworks.com/parallel-computing). Also, scoreAllSpectra_spmd has also been validated using $\geq$100 workers in a computer cluster that was equipped with the MATLAB distributed computing server (www.mathworks.com/products/distriben/). This considerably reduces computing time for large datasets.

Glycan format transformation

• **linucs2SmallGlyPep**	- Convert from LINUCS to SmallGlyPep format

Classes

• **Chemele**	- An object of chemical element
• **Glycan**	- An object of glycan

• **Chemformula**	- An object of chemical formula
• **AminoAcid**	- An object of amino acid
• **Protease**	- An object of protease
• **Modification**	- An object of protein modification
• **mzXML**	- An object of mzXML file

8. Acknowledgements

We thank Seattle Proteomics Center for providing JRAP, a Java library for reading MS data from mzXML files (http://tools.proteomecenter.org/wiki/index.php?title=Software:JRAP) and the Proteome Informatics group at Switzerland for their JRAP extension work (http://javaprotlib.sourceforge.net/packages/io/jrap/).