

BiMat Analysis GUI Guide

Carson Kennedy, Samuel Blair, Garrett Hairston, Dr. Astrid Layton*

J. Mike Walker '66 Dept. of Mechanical Engineering, Texas A&M University, *alayton@tamu.edu

Last Update: July 3, 2022

System Requirements and Expectations

- MATLAB® 2021a. This GUI may work in previous or newer versions but was not tested on them.
 - Verify that the version of MATLAB being used has the AppDesigner Library
 - It is a common error within this program to be missing the `nansum()` function. If this happens, it is recommended that MATLAB is reinstalled.
- The user is expected to have basic MATLAB® knowledge.
- The user is also expected to have a basic understanding of the Bipartite analysis being done, such that the vocabulary used is not foreign.
- If the user has any questions regarding the GUI and its functions, then contact Carson Kennedy at cmk012000@gmail.com
- Read the “BiMat: Start Guide” within the “BiMat-Master” Folder of the program for an extensive documentation for the BiMat Library.
- Visit the BiMat git hub at <https://bimat.github.io/> for even more documentation regarding the BiMat Library

Introduction

A bipartite network is a representation of the interactions between the column and row nodes. A value of one or zero is used to quantitatively represent the interactions in a matrix form, where a one represents an interaction and a zero represents no interaction. See Figure 1 for an example of how a bipartite matrix is created. In the future it is a goal to be able to use magnitudes within the bipartite analysis, but this GUI and the BiMat Library cannot currently analyze a bipartite matrix with magnitudes.

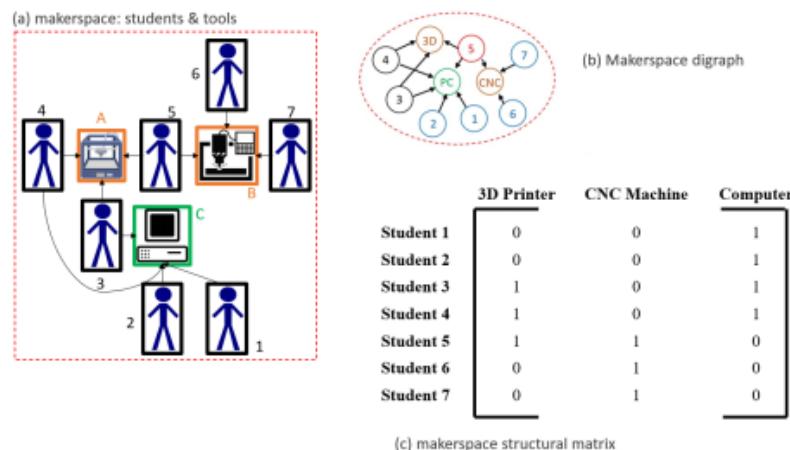


Figure 1: Bipartite network analysis for tools and student interactions [1].

The modularity analysis consists of two major steps: Partitioning the modules and calculating the P-value and z-score. Using the methods detailed in “*Using A Modularity Analysis to Determine Tool and Student Roles within Makerspaces*” [1], the MATLAB package BiMat allows for the corresponding analysis of a bipartite matrix. One important part of this document is the explanation of how BiMat can use a few different modularity algorithms that define the modules and the standard modularity metric. This paper also explains how the P and Z values are defined, as well as an example as to how the modularity is determined using the Leading Eigenvector Method for optimization.

The BiMat software used within this GUI was developed by: Cesar O. Flores, Timothee Poisot, Sergi Valverde, and Joshua S. Weitz. “BiMat: a MATLAB(R) package to facilitate the analysis of bipartite networks.”

Algorithms in BiMat were first developed in the following publications:

- NODF algorithm: Mario Almeida-Neto, Paulo Guimaraes, Paulo R. Guimaraes Jr, Rafael D. Loyola, and Werner Ulrich. “A consistent metric for nestedness analysis in ecological systems: reconciling concept and measurement.” *Oikos*, 117(8):1227-1239, 2008.
- NTC algorithm: Wirt Atmar and Bruce D. Patterson. “The measure of order and disorder in the distribution of species in fragmented habitat.” *Oecologia*, 96:373-382, 1993.
- Adaptive Brim Algorithm: Michael Barber. “Modularity and community detection in bipartite networks.” *Physical Review E*, 76:066102, 2007.
- LP&Brim algorithm: Xin Liu and Tsuyoshi Murata. “Community Detection in Large-Scale Bipartite Networks.” *Proceedings of the International Conference on Web Intelligence and Intelligent Agent Technology*, 1: 50-57, 2009.
- Leading Eigenvector Algorithm: M. E. J. Newman. “Modularity and community structure in networks.” *Proc Natl Acad Sci*, 103: 8577-8582, 2006.

To take the modularity analysis one step further the nodes are assigned a P-value and z-score. The Z-score quantifies the intra-module degree of interactions. For example, if many students that use the same set of tools are also all using the laser cutter, the laser cutter would have a high connectivity value. Whereas the P-Value quantifies the intra-module degree of interactions. For example, two nodes with the same z-score will probably play different roles if one of them is connected to several nodes in other modules while the other is not. From there the two degrees of interactions can be placed onto a scatter plot that groups them into seven different roles: Non-Hubs - Ultra-peripheral nodes ($P \approx 0.02$), Peripheral nodes ($0.02 > P < 0.625$), Non-hub connectors ($0.625 \geq P < 0.8$), Non-hub kinless nodes ($P > 0.8$) and Hubs - Provincial hubs ($P < 0.3$), Connector hubs ($0.3 \geq P < 0.75$), Kinless hubs ($P > 0.75$). Below this scatter plot and the different regions can be seen in Figure 2. See “*Using A Modularity Analysis to Determine Tool and Student Roles within Makerspaces*” [1] and “*Cartography of complex networks: modules and universal roles*” [4] for a detailed description of how to obtain the P and Z values as well as how to define the roles and their importance.

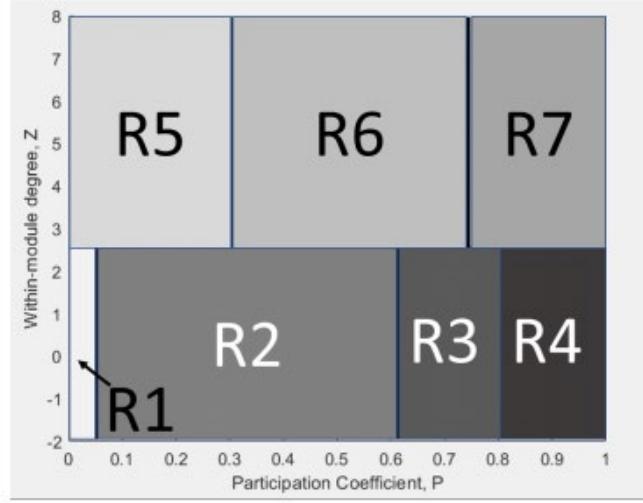


Figure 2: Modularity analysis sectioning determined by connectivity (z) and participation (p) values. Figure from [1].

"For networks of species interactions, nestedness occurs when specialist species tend to interact with proper subsets of the species that interact with more generalist species [5]." This paper explains how the NODF method calculates the Nestedness for the rows, column, and the collective matrix. Calculating this nestedness also requires that the bipartite matrix is optimized to be that the columns with the largest sum should be placed furthest to the left, and that the row with the largest sum are placed closest to the Top. The NODF method then splits the column and rows up into pairs and calculates their nestedness. From there, the nestedness of the column and row pairs were averaged to create the average nestedness amongst rows and columns. Then the two can be averaged into the nestedness metric for the collective matrix.

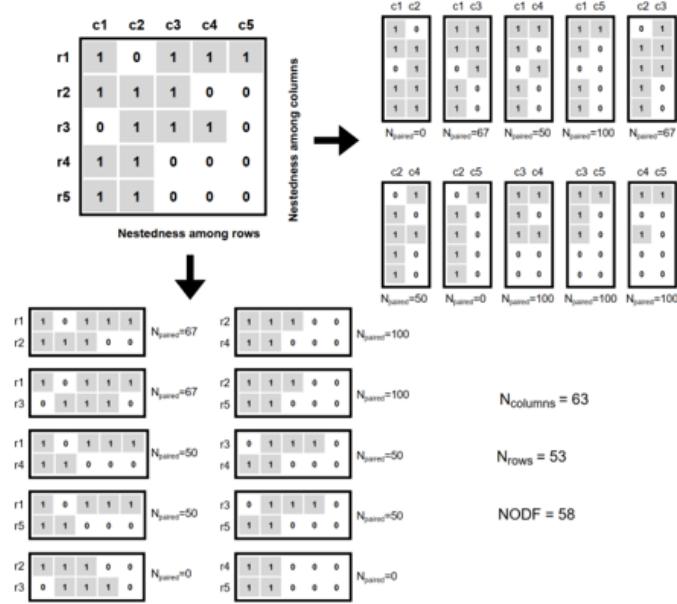


Figure 3: Illustration of the way by which nestedness is quantified according to NODF [2].

Summary of the Guide

The purpose of this guide is to assist in the installation and operation of the GUI, as well as provide a summary of each individual function within the program. The GUI is a visual way of running the BiMat analysis on nestedness and modularity of bipartite networks. These networks can be entered manually into the program by using the Input Step (A.2.3) or by creating a new preformatted sheet with Section (A.4). For a detailed description of how the BiMat analyzes the data see the extended documentation of the BiMat library on: <http://bimat.github.io/> as well as the BiMat Start Guide within the BiMat-Master folder. This is also the folder that contains the scripts that run the analysis and used for variable definition or operations within this GUI.

The results of this analysis are shown in many ways throughout the program. The most important results are featured in the Tabs window on the right side of the GUI. These results are split into 4 major tabs and a fifth used for comparison. The General properties tab shows the basic information about the bipartite matrix that is being analyzed like its size, connectance, and number of interactions.

Next, the Modularity tab displays the general modularity statistics like Number of modules and the Standard Modularity metric for whichever algorithm is selected. These metrics are defined using the “detect” function within the “LPBrim.m”, “LeadingEigenVector.m”, or “AdaptiveBrim.m” script in combination with the “detect” function within the “BipartiteModularity” script. The contents of these modules themselves can also be shown using the “Show Modules” button on this tab. The button opens a separate window that allows for a module to be selected and shown.

The third tab is the P and Z tab, which is the most complex as it shows the general results within the tab and open the separate P and Z page. The information shown on this tab is the name, p value, z score, and role for each individual element. Specifically for the rows or columns exclusively because the P and Z values are defined using only the rows or only the columns. The “Modules Switch” is used to toggle between the row and column elements. Clicking on the Plot Button opens the P and Z page in a separate window. This window shows the same elements as the original P and Z tab but ordered into columns based on their roles, as well as plots each element on a scatter plot based on their P and Z values. This plot shows how the roles are defined by grouping the P and Z values.

The Nestedness tab shows the nestedness values for the rows, columns, and the collective matrix. These metrics are defined using the “detect” function within the “Nestedness.m” script. It is also possible to view the nested matrix on this tab by using the “View Nested Matrix” button. This shows the bipartite matrix rearranged into the most nested matrix possible using the “SORT_MATRIX” function within the “MatrixFunctions” script.

The last tab is the Compare tab this tab allows for a quick comparison between different bipartite matrices by displaying the results from the major four tabs in one column. This table is only cleared once the program is closed.

The GUI is also capable of displaying four different graphs from the “PlotWebs.m” function with their prospective functions. These graphs include the Modular Graph, Matrix, Modular Matrix, and Nested Matrix. They can be viewed one at a time with the generate tab or all at once with the view all graphs button, which requires that the “Export” button has been clicked with the “Export Plots” checkbox checked.

To use these results outside of the GUI, the Export Panel can be used to save all the results to a specified excel sheet and tab. This export includes all the results from the main four tabs being saved in the excel sheet, as well as saves an image of the four plots in the “Graphs” folder within the “Results” folder.

References

- 1) S. E. Blair, & Banks, H. D., & Linsey, J. S., & Layton, A., "Modularity Analysis of Makerspaces to Determine Potential Hubs and Critical Tools in the Makerspace," ASEE Virtual Annual Conference, 2022.
- 2) S. E. Blair, & Banks, H. D., & Linsey, J. S., & Layton, A., "Bipartite Network Analysis Utilizing Survey Data to Determine Student and Tool Interactions in a Makerspace," ASEE Virtual Annual Conference, 2021.
- 3) C. Brehm, J. Linsey, and A. Layton, "Using a Modularity Analysis to Determine Tool and Student Roles within Makerspaces," presented at the 2020 ASEE Virtual Annual Conference, Virtual Online, June 22-26, 2020, 29636
- 4) R. Guimerà and L. A. N. Amaral, "Cartography of complex networks: modules and universal roles," Journal of Statistical Mechanics: Theory and Experiment, vol. 2005, no. 02, p. P02001, 2005/02/02 2005, doi: 10.1088/1742-5468/2005/02/p02001
- 5) Almeida-Neto, M., Guimarães, P., Guimarães, P.R., Jr, Loyola, R.D. and Ulrich, W. (2008), A consistent metric for nestedness analysis in ecological systems: reconciling concept and measurement. Oikos, 117: 1227-1239.
- 6) Cesar O. Flores, Timothee Poisot, Sergi Valverde, and Joshua S. Weitz. "BiMat: a MATLAB(R) package to facilitate the analysis of bipartite networks."
- 7) Wirt Atmar and Bruce D. Patterson. "The measure of order and disorder in the distribution of species in fragmented habitat." Oecologia, 96:373-382, 1993.
- 8) Michael Barber. "Modularity and community detection in bipartite networks." Physical Review E, 76:066102, 2007.
- 9) Xin Liu and Tsuyoshi Murata. "Community Detection in Large-Scale Bipartite Networks." Proceedings of the International Conference on Web Intelligence and Intelligent Agent Technology, 1: 50-57, 2009
- 10) M. E. J. Newman. "Modularity and community structure in networks." Proc Natl Acad Sci, 103: 8577-8582, 2006.

Table of Contents

BiMat Analysis GUI Guide	1
System Requirements and Expectations	1
Introduction	1
Summary of the Guide	3
References	5
Table of Contents.....	6
A: Program Operation.....	10
1 Installing App	10
2 Running the Program.....	12
2.1 Selecting Defaults.....	13
2.2 File Selection	13
2.2.1 Manual Entry.....	13
2.2.2 Preformatted Sheet	14
2.3 Matrix Input	14
2.3.1 Manual Entry.....	14
2.3.2 Preformatted sheet.....	15
2.3.3 Verify Entry.....	15
2.4 Run Analysis	15
2.5 Generating Plots.....	16
2.5.1 View a Single Plots	16
2.5.2 Show all Plots	16
2.6 Modularity.....	17
2.6.1 Algorithm Selection.....	17
2.6.2 Viewing Modules.....	17
2.7 P and Z	18
2.7.1 P and Z Page	20
2.8 Nestedness.....	21
2.8.1 Viewing Nested Matrix.....	21
2.9 Compare.....	22
2.10 Export Results	22
2.10.1 Export Data to Results Sheet.....	22

2.10.2	Show all Plots	23
3	Adding a Data Sheet.....	24
4	Adding a Preformatted Sheet	24
4.1	File Location	24
4.2	Matrix Properties Script.....	25
4.3	File Drop Down.....	26
4.4	Generate Tab Options Function.....	27
4.5	Construct Premade Matrix Function.....	28
5	Adding a Results Sheet.....	28
6	Adding a Default	28
6.1	Adding the Dropdown Value.....	28
6.2	Editing the Options Drop Down Value Changed callback(B.1.3.2)	30
-	30
B:	Program Description	31
1	Main Page	31
1.1	Analysis Functions.....	31
1.1.1	File Path.....	31
1.1.2	Define Matrix	32
1.1.3	Define Bipartite Matrix	33
1.1.4	General Properties	34
1.1.5	Modularity.....	35
1.1.6	Nestedness.....	36
1.1.7	NestedMatrix	37
1.1.8	P and Z.....	39
1.1.9	Define Roles	44
1.2	GUI Functions.....	45
1.2.1	Run BiMat.....	45
1.2.2	Generate Plots.....	46
1.2.3	Generate Tab Options.....	46
1.2.4	Construct Premade Matrix.....	47
1.2.5	Fill P and Z	48
1.2.6	Launch P and Z	49
1.2.7	Export to Excel	50

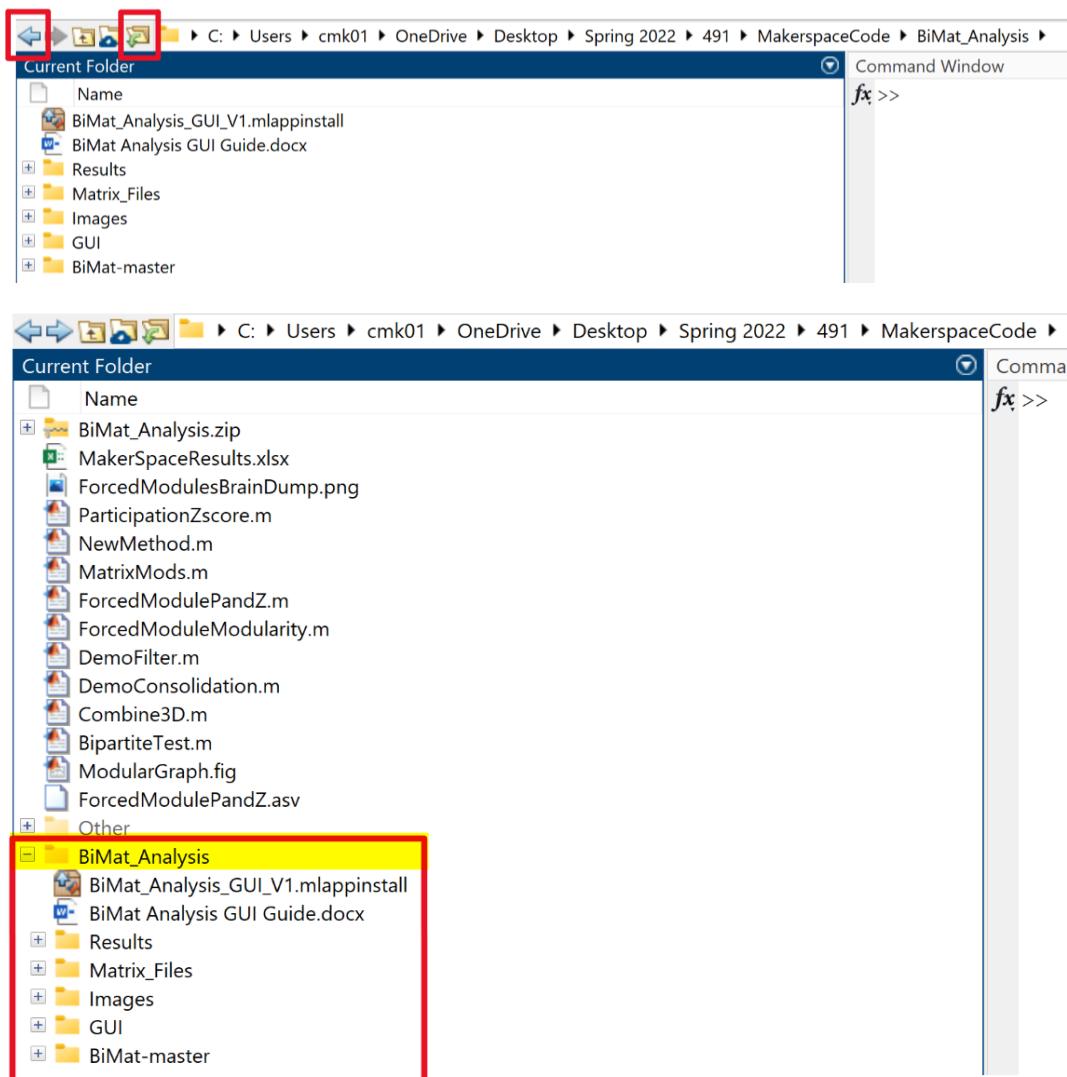
1.3	Callbacks.....	52
1.3.1	Startup Function.....	52
1.3.2	Option Drop Down Value Changed.....	53
1.3.3	Update Button Pushed.....	53
1.3.4	Edit Defaults Button Pushed	53
1.3.5	Use a Preformatted Sheet Button Pushed.....	54
1.3.6	File Dropdown Value Changed.....	54
1.3.7	Tab Dropdown Value Changed	54
1.3.8	View Matrix Button Pushed	55
1.3.9	List Tabs Button Pushed.....	55
1.3.10	Tabs Check Box Value Changed	56
1.3.11	File Name Value Changing	56
1.3.12	File Name Value Changed	56
1.3.13	Run Button Pushed	57
1.3.14	Algorithm Drop Down Value Changed.....	58
1.3.15	Show Modules Button Pushed.....	58
1.3.16	Modules Switch Value Changed.....	59
1.3.17	Plot Button Pushed	59
1.3.18	View Nested Matrix Button Pushed.....	60
1.3.19	Add Data Button Pushed.....	61
1.3.20	Generate Button Pushed.....	62
1.3.21	Export Button Pushed	62
1.3.22	Show all Plots Button Pushed	64
2	Modules Page.....	64
2.1	Functions.....	66
2.1.1	Fill Matrix	66
2.2	Callbacks.....	67
2.2.1	Startup Function.....	67
2.2.2	Module Spinner Value Changed.....	68
2.2.3	Show Matrix Button Pushed	68
2.2.4	View Module Size Button Pushed	70
3	P and Z Page	71
3.1	Functions.....	72

3.1.1	Plot P and Z	72
3.1.2	Reset Plot Labels	74
3.2	Callbacks.....	75
3.2.1	Startup Function.....	75
3.2.2	Roles Table Cell Selection.....	76
3.2.3	Reset Button Pushed.....	77
3.2.4	Export Screenshot Button Pushed	77
4	Plots Page.....	78
4.1	Callbacks.....	80
4.1.1	Startup Function.....	80
5	Matrix Page	80
5.1	Functions.....	81
5.1.1	Fill Figures	81
5.2	Callbacks.....	82
5.2.1	Startup Function.....	82
6	List Page	82
6.1	Callbacks.....	83
6.1.1	Startup Function.....	83
7	*Name*MatrixProperties.m	83
7.1	Functions.....	83
7.1.1	*Name*MatrixProperties	84

A: Program Operation

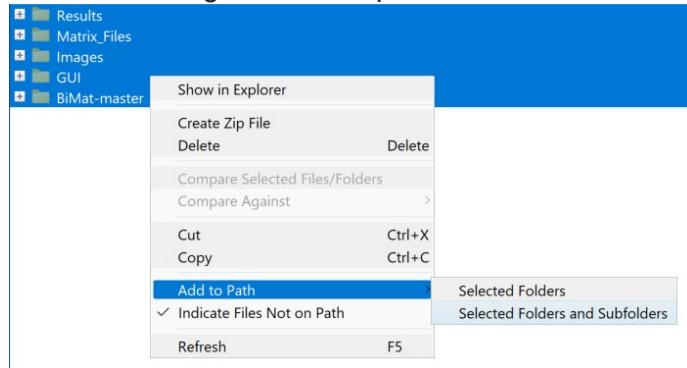
1 Installing App

- I. Download and Unzip the “BiMat_Analysis” folder to a specific location on your computer
- II. Open this folder in your MATLAB directory. Not the actual “BiMat_Analysis” folder, specifically the folder that you put it in. Otherwise, the export functions will not be able to locate their destination
 - a. The open folder button is highlighted in the first image below.
 - b. If you open the “BiMat_Analysis” folder you can use the back arrow to return to its location. The directory is the second image below can be used as an example, whereas the first image is incorrect and highlights the back arrow.

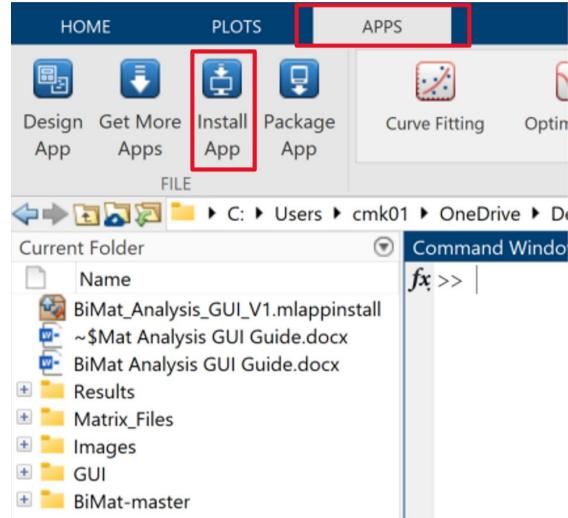


- III. Make sure that all of the folders and files within "BiMat_Analysis" are added to your path
 - a. Select All of the Folders
 - b. Right Click on any of the selected folders

- c. Select “Add to Path” and then “Selected Folders and Subfolders”
- d. The selected files should go from transparent to solid



- IV. Locate the and Click the “Install App” button on the “Apps” tab at the top of the screen



- V. Select and Open the “BiMat_Analysis_GUI_V1.mlappinstall” file within the folder that you just unzipped

BiMat-master	4/20/2022 8:56 PM	File folder
GUI	4/21/2022 2:03 PM	File folder
Images	3/27/2022 2:43 PM	File folder
Matrix_Files	4/12/2022 9:28 PM	File folder
Results	4/21/2022 9:01 AM	File folder
BiMat_Analysis_GUI_V1.mlappinstall	4/12/2022 8:31 PM	MATLAB App Inst... 462 KB

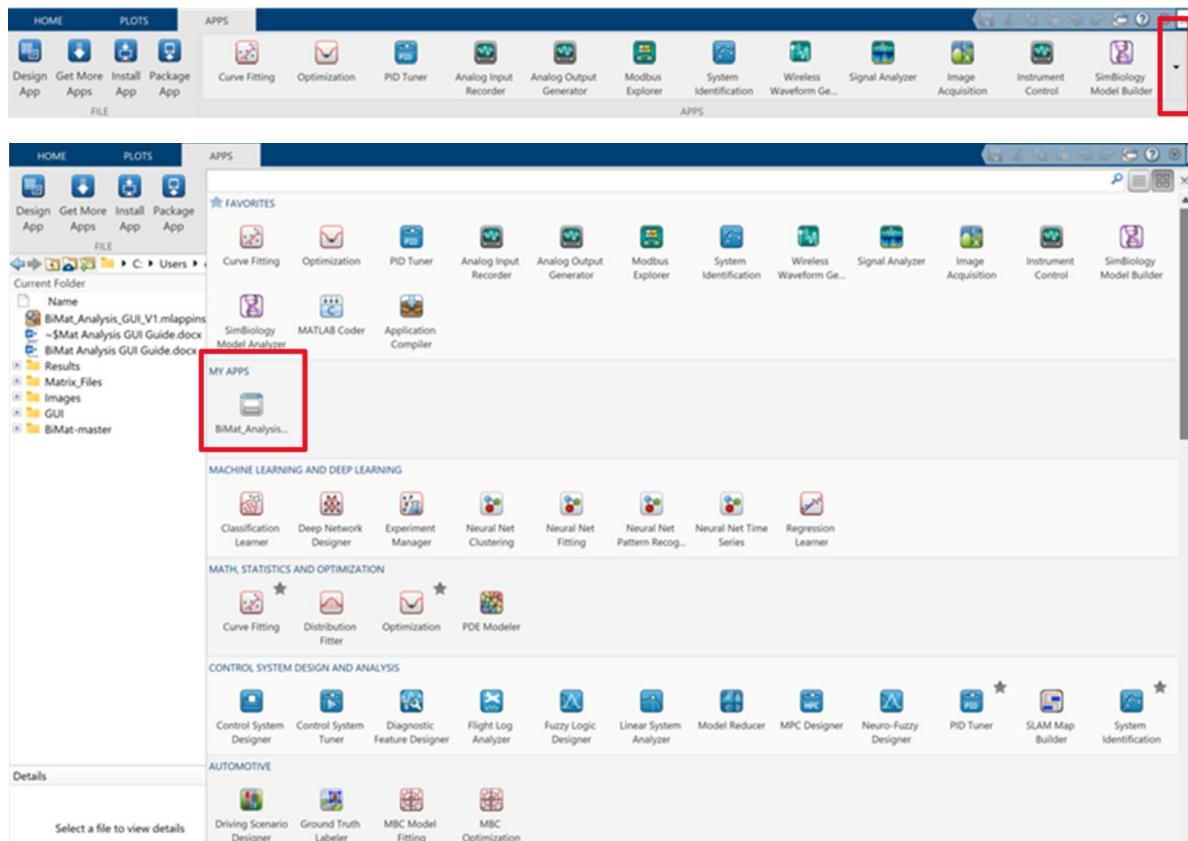


- VI. Click Install



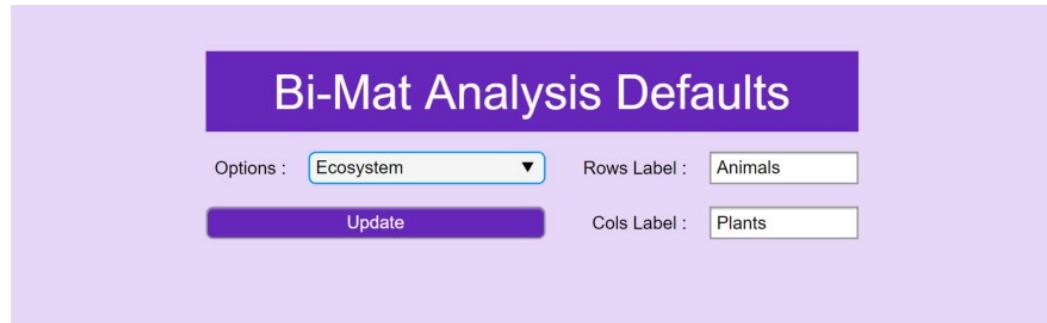
VII. Now you can run the GUI by selecting it in the Apps tab

- If it is not visible select the arrow opening the drop down to the right of all of your apps. It should be in the "My Apps" row.



2 Running the Program

2.1 Selecting Defaults



- I. Select the Default Row and Column Labels by using the Options Dropdown
 - a. If Custom is Chosen, then the Row and Column labels should be entered manually
 - b. Activates Callback B.1.3.2 – Option Dropdown Value Changed
- II. Click the Update Button
 - a. Activates Callback B.1.3.3 – Update Button Pushed

2.2 File Selection

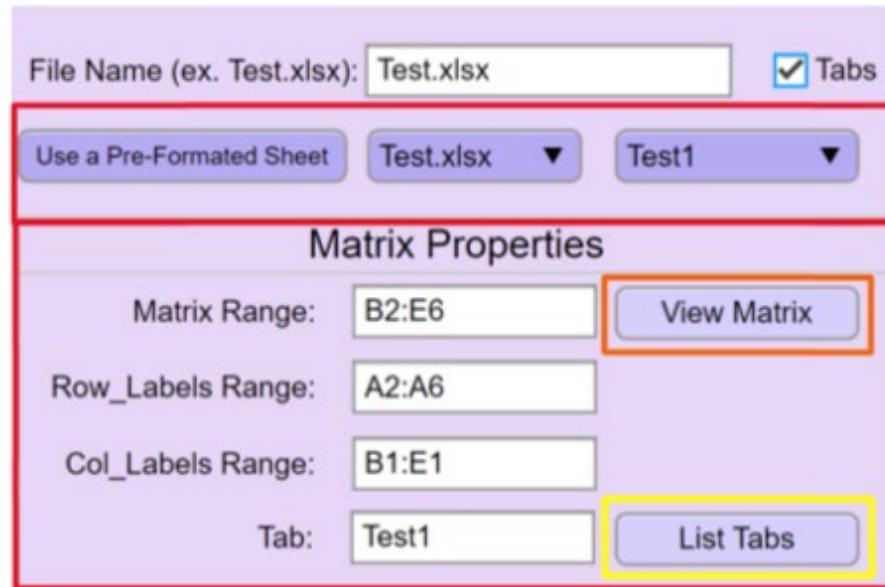
2.2.1 Manual Entry

- I. Type the filename into the File Name blank
 - a. Don't Forget to include the file extension
 - b. Make sure the file is in the "Matrix_Files" folder. (See section A.2 for a detailed description of adding a sheet.)
 - c. Activates Callback B.1.3.11 – File Name Value Changing
 - d. Activates Callback B.1.3.12 – File Name Value Changed
- II. Check the Tab Box if you are going to be using a specific Tab
 - a. Activates Callback B.1.3.10 – Tabs Check Box Value Changed

2.2.2 Preformatted Sheet

- I. Look at Section A.3 for Inputting a Preformatted Sheet
- II. Select the Preformatted Sheet Using the File Drop Down
 - a. Activates Callback B.1.3.6 – File Dropdown Value Changed
- III. Select the Tab using the Tab's Dropdown
 - a. Activates Callback B.1.3.7 – Tab Dropdown Value Changed
- IV. Click the Use a Pre-Formatted Sheet Button
 - a. Activates Callback B.1.3.5 – Use a Pre-Formatted Sheet Button Pushed

2.3 Matrix Input



2.3.1 Manual Entry

- I. Input the Matrix Range
 - a. The Matrix Range is the cells that contain the matrix data
- II. Input the Row Labels Range
 - a. The Row Labels Range is the cells that contain the row names
- III. Input the Column Label Range
 - a. The Column Labels Range is the cells that contain the Column names
- IV. Make sure that all of the ranges are specified using a colon not a semi-colon
- V. Input the Tab Name within the Sheet that the program is using
 - a. This only shows up when the Tabs Checkbox is checked
 - b. You can use the List Tabs Button to get a list of Tabs that are available (Yellow Box)
 - i. Activates Callback B.1.3.9 – List Tabs Button Pushed

GTF19
GTF20
TAMUF20
TAMUF20_V2
TAMUS21
TAMUS21Forced
TAMUS21_G
TAMUS21_W
GTS21
GTS21Forced
GTS21_W
GT21_G
GTF19_ToolM
2021 TAMU Adjusted
2021 GT Adjusted
2020 TAMU Adjusted

2.3.2 Preformatted sheet

- I. Look at Section A.3 for Inputting a Preformatted Sheet
- II. Select the Preformatted Sheet Using the File Drop Down
 - a. Activates Callback B.1.3.6 – File Dropdown Value Changed
- III. Select the Tab using the Tab's Dropdown
 - a. Activates Callback B.1.3.7 – Tab Dropdown Value Changed
- IV. Click the Use a Pre-Formatted Sheet Button
 - a. Activates Callback B.1.3.5 – Use a Pre-Formatted Sheet Button Pushed

2.3.3 Verify Entry

- I. Click the View Matrix Button to see how the Matrix and its Labels were inputted
 - a. Activates Callback B.1.3.8 – View Matrix Button Pushed

Test.xlsx - Test1

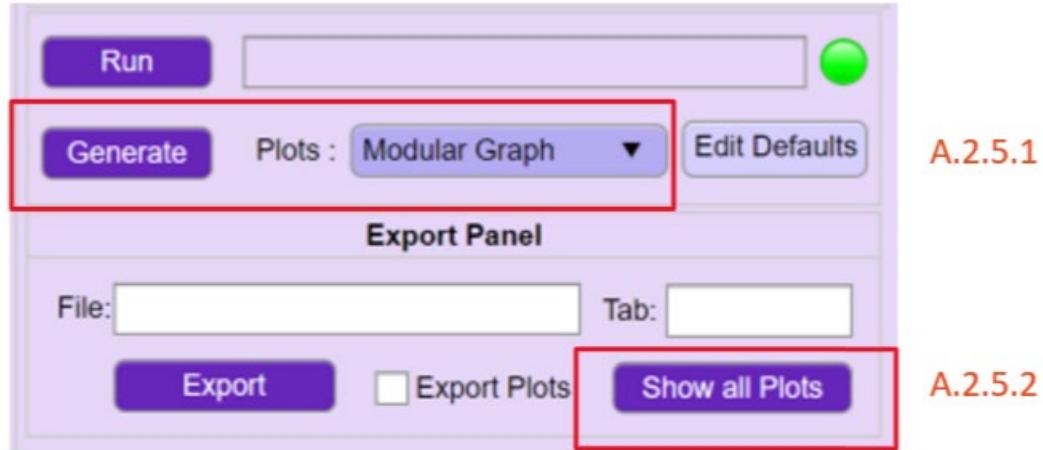
	flower 1	flower 2	grass 1	grass 2	
insect 1	1	0	1	1	1
insect 2	1	1	1	1	1
insect 3	1	0	0	0	1
bird 1	0	1	1	1	1
bird 2	0	0	1	0	

2.4 Run Analysis

- I. Click the Run Button

- II. Verify that the steps above are done correctly. If not, an error message will be returned

2.5 Generating Plots



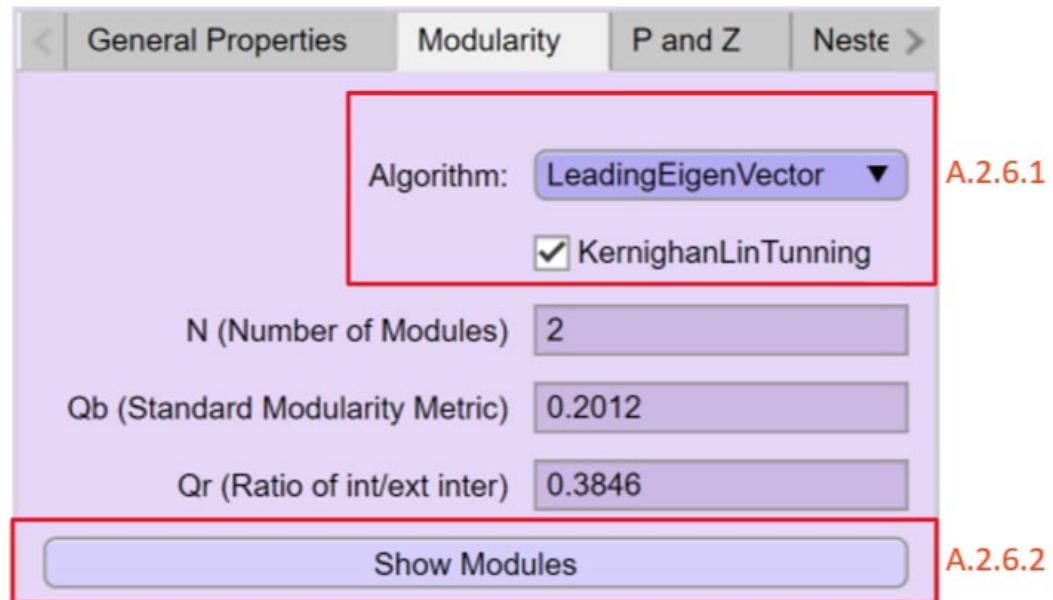
2.5.1 View a Single Plot

- I. Select the Graph that you would like to view in the Plot Dropdown
- II. Click the Generate Button
 - a. Activates Callback B.1.3.20 – Generate Button Pushed

2.5.2 Show all Plots

- I. Click the Show all Plots Button
 - a. This requires that the export button has been pushed and doesn't show unless the export plots checkbox is checked
 - i. If the export has not been completed then the graphs shown will be from the previous time the program was ran, even though the label will be updated
 - b. See section A.1.9 for the Export steps
 - c. Activates the Callback B.1.3.22 – Show all Plots Button Pushed

2.6 Modularity



2.6.1 Algorithm Selection

- I. Select the Algorithm that you would like to use in the Algorithm Dropdown
 - a. The KerninghanLin Checkbox Only shows when LeadingEigenVector is chosen
 - b. Activates Callback B.1.3.14 – Algorithm Drop Down Value Changed

2.6.2 Viewing Modules

- I. Click the Show Modules Button
 - a. Activates Callback B.1.3.15 – Show Modules Button Pushed
- II. Use the steps below in A.1.5.2.1 and A.1.5.2.2 to select the module you would like to view
- III. Click the Show Matrix Button (yellow box)
 - a. Activates Callback B.2.2.3 – Show Matrix Button Pushed

Modules			
	flower 2	grass 1	
insect 2		1	1
bird 1		1	1
bird 2		0	1

Show Matrix

Module:

View Module Sizes

A.2.6.2.1

A.2.6.2.2

2.6.2.1 Module Selection

- I. Select the Module using the spinner
 - a. Activates Callback B.2.2.2 – Module Spinner Value Changed
 - b. If you select a number larger than the number of modules an error will be returned

2.6.2.2 Module Sizes

- I. Click the View Module Sizes Button if you would like to see the number of modules and their sizes
 - a. Activates Callback B.2.2.4 – View Module Sizes Button Pushed
 - b. If you select a number larger than the number of modules an error will be returned

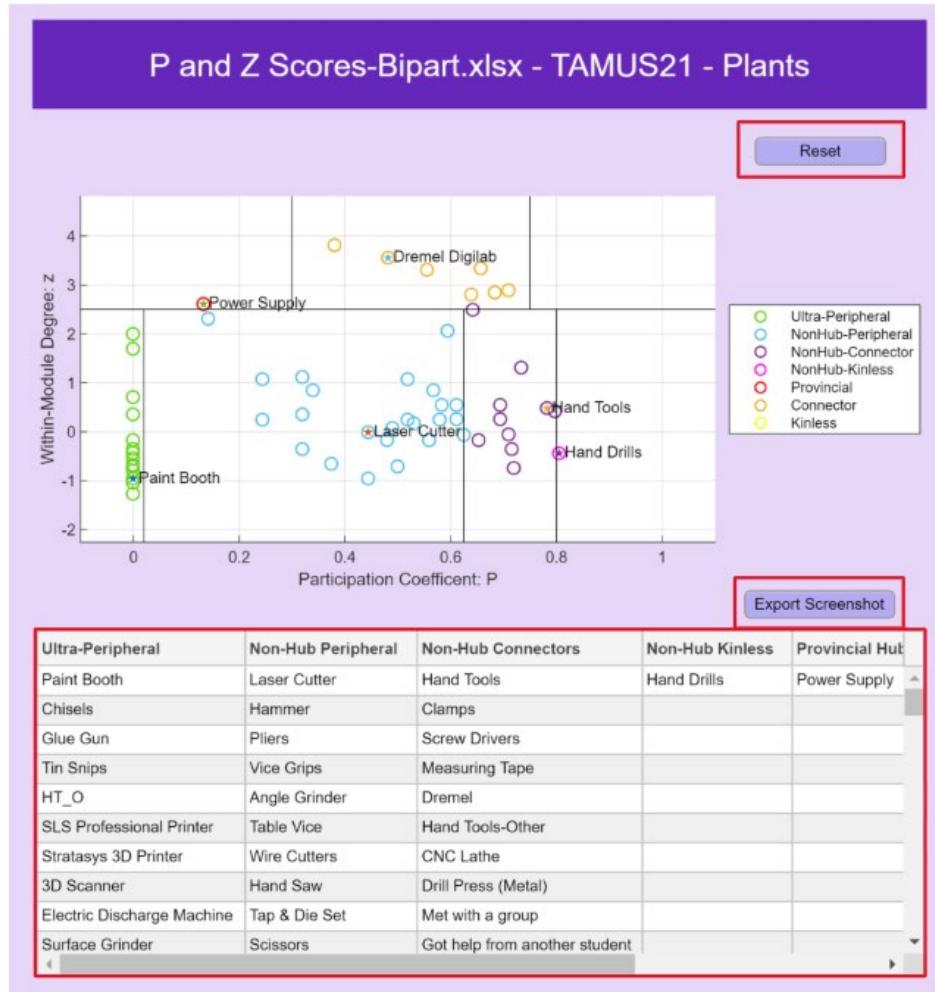
2.7 P and Z

General Properties Modularity P and Z Nested

Plants	P-Value	z-Value	Role
Clamps	0.71605	-0.3511	Non-Hub Conn
Screw Drivers	0.79688	0.42875	Non-Hub Conn
Hand Drills	0.80556	-0.43118	Non-Hub Kinle
Angle Grinder	0.375	-0.65344	Non-Hub Perip
Chisels	0	-0.79057	Ultra-Periphera
Measuring Tape	0.69388	0.55592	Non-Hub Conn
Table Vice	0.34	0.85826	Non-Hub Perip
Glue Gun	0	-0.54308	Ultra-Periphera
Wire Cutters	0.53125	0.17878	Non-Hub Perip
Hand Saw	0.625	-0.0571...	Non-Hub Perip
Dremel	0.69388	0.26352	Non-Hub Conn
Tap & Die Set	0.51852	0.25358	Non-Hub Perip

Rows
 Columns

- I. The P and Z functions are for a Unipartite Function so you must select whether you would like to analyze the row or column elements. This is done with the Modules Switch
 - a. Activates Callback B.1.3.16 – Modules Switch Value Changed
- II. Click the Plot button to view a more detailed view of the P and Z analysis
 - a. Activates Callback B.1.3.17 – Plot Button Pushed



2.7.1 P and Z Page

2.7.1.1 Cell Selection

- Click on any of the cells with text within the Roles table
 - Activates Callback B.3.2.2 – Roles Table Cell Selection

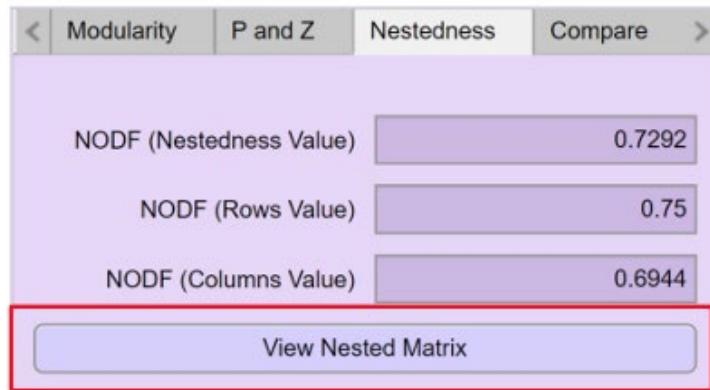
2.7.1.2 Reset Labels

- Click on the Reset Button to remove any of the labels created by the Cell selections
 - Activates Callback B.3.2.3 – Reset Button Pushed

2.7.1.3 Export Screenshot

- Click on the Export Screenshot Button to save an image of the window including any of the labels created by the Cell selections to the “P and Z” folder within the “Results” folder
 - Activates Callback B.3.2.3 – Reset Button Pushed

2.8 Nestedness



A.2.8.1

2.8.1 Viewing Nested Matrix

- II. Click the View Nested Matrix
 - a. Activates Callback B.1.3.18 – View Nested Matrix Button Pushed

The figure shows a screenshot of a Microsoft Excel spreadsheet titled "Test.xlsx - Test1". The spreadsheet contains a single table with the following data:

	grass 1	grass 2	flower 1	flower 2
insect 2	1	1	1	1
insect 1	1	1	1	0
bird 1	1	1	0	1
insect 3	0	1	1	0
bird 2	1	0	0	0

2.9 Compare

Compare	
General Properties	Test1
Number of Elements	9.0000
Number of Row Elements	5.0000
Number of Col Elements	4.0000
Number of Interactions	13.0000
Size	20.0000
Connectance or Fill	0.6500
Modularity	
Algorithm	LeadingEige...
N - Number of Modules	2.0000
Qb - Standard Modularity Metric	0.2012
Qr - Ratio of int/ext interactions	0.3846
Nestedness	
NODF Nestedness Value	0.7292
NODF Rows Value	0.7500
NODF Columns Value	0.6944

Add Data

- I. For a quick comparison between two different tabs or sheets, the comparison tab uses the Add Data button to show the results of each analysis in separate columns.
 - a. Activates Callback B.1.3.19 – Add Data Button Pushed

2.10 Export Results

Export Panel

File: EcosystemResults.xlsx Tab: GTS21

Export Plots

A.2.10.1

A.2.10.2

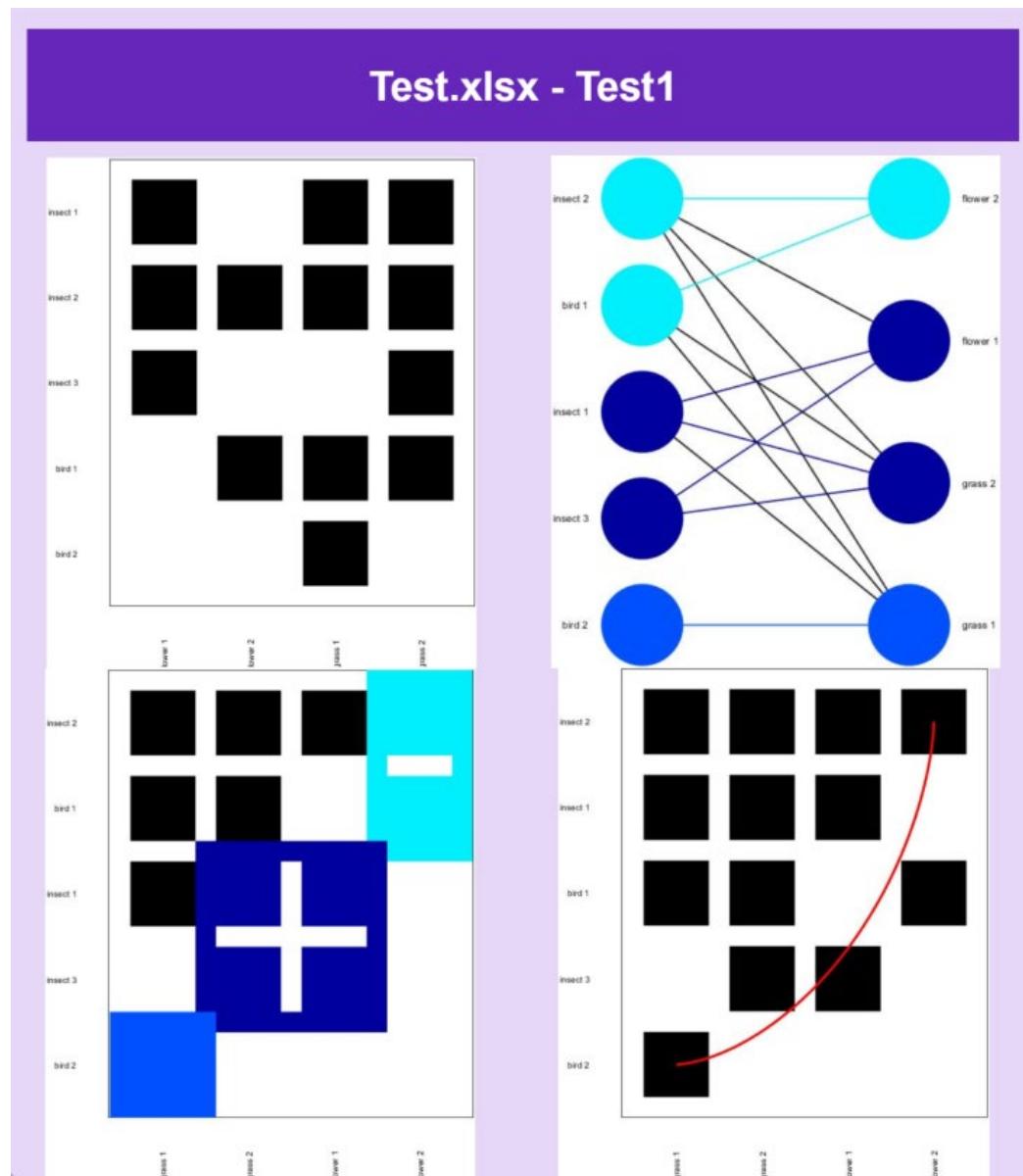
2.10.1 Export Data to Results Sheet

- I. Fill in the File and Tab Blanks with the File name that the results should be exported to.
 - a. Make sure to include the file extension in the filename
 - b. Refer to Section A.4 for the results file location and the steps to add one

- II. Check the Export Plots checkbox if you would like to export the 4 plots into the “Graphs” within the “Results” folder
 - a. This is required for the Show all Plots button to be visible
- III. Click the Exports button
 - a. This is also required for the Show all Plots button to be visible

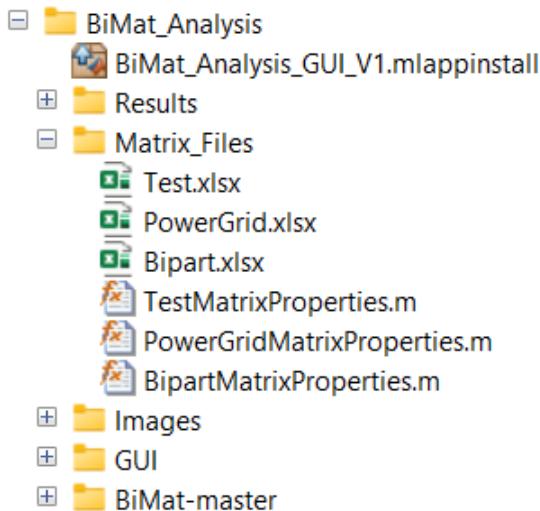
2.10.2 Show all Plots

- I. Click the Show all Plots Button
 - a. Activates Callback B.1.3.22 – Show all Plots Button Pushed



3 Adding a Data Sheet

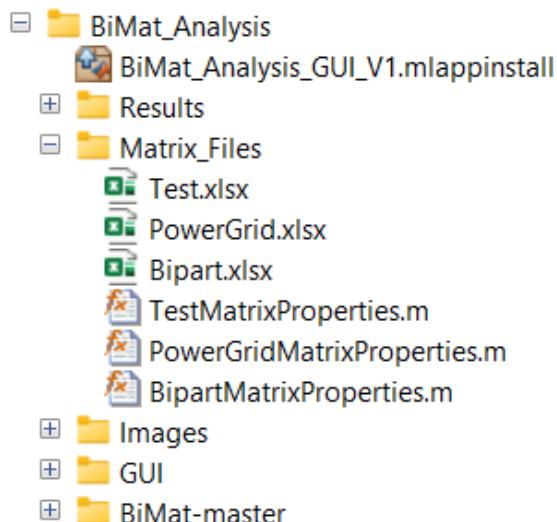
To add a data sheet, the process is simply adding a excel sheet to the “Matrix_Files” folder within “BiMat_Analysis”. If the user would like this sheet to be in the file and tabs dropdowns, then use “Adding a Preformatted Sheet” (A.4) If not then the ranges will need to be inputted manually.



4 Adding a Preformatted Sheet

4.1 File Location

To add a data sheet, the process is simply adding a excel sheet to the “Matrix_Files” folder within “BiMat_Analysis”.



4.2 Matrix Properties Script

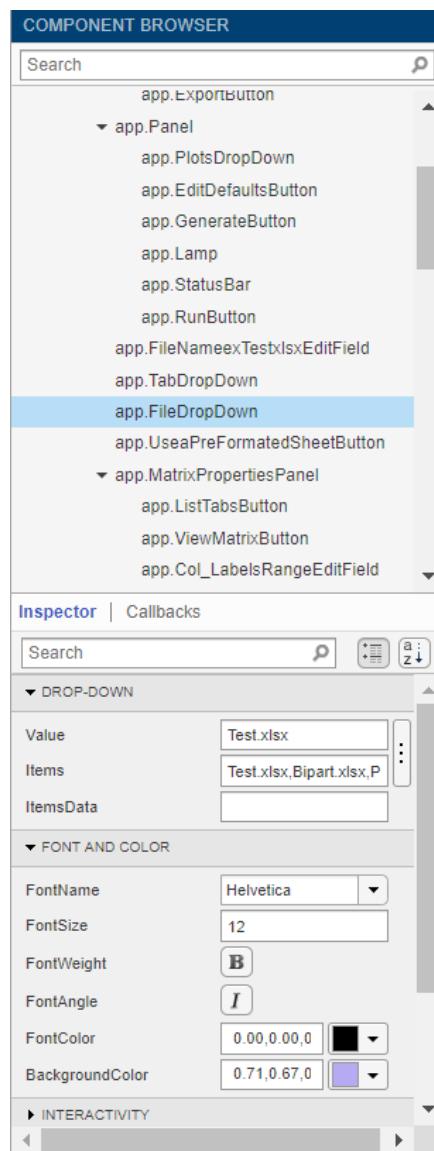
This sheet replaces the step of adding the range values to the matrix properties panel, by having them defined in a separate MATLAB script. To create a new one its easiest to use the “TestMatrixProperties.M” file as an example. Also see the *Name*MatrixProperties section (B.6). A detailed list of the steps to create a similar script are below.

- I. Create a new MATLAB sheet and paste the “TestMatrixProperties.m” script into this sheet
 - a. Name this file using the notation “*Name*MatrixProperties.m”.
- II. Replace the function name with the same name as the file
- III. Copy the elseif statement and paste it below the previous one
 - a. Change the original if and elseif statements using the steps below
 - b. Change the string value within the strcmp() function to represent the name of each tab
 - c. Change the range values to represent the cell locations of the Row and Column labels as well as the matrix data

```
1  function [matrix_range, row_labels_range, col_labels_range, tabs] = TestMatrixProperties(tab_name, file)
2  raw_tabs = sheetnames(file);
3  %accepted_tabs = {};
4  tabs = raw_tabs;
5  %disp(tabs);
6
7  if strcmp(tab_name, 'Test1') == true
8      matrix_range = 'B2:E6';
9      row_labels_range = 'A2:A6';
10     col_labels_range = 'B1:E1';
11  elseif strcmp(tab_name, 'Test2') == true
12      matrix_range = 'B2:E6';
13      row_labels_range = 'A2:A6';
14      col_labels_range = 'B1:E1';
15  else
16  end
17  end
```

4.3 File Drop Down

- I. Use the Component designer within AppDesigner to locate and select the “app.FileDropDown”.
- II. Once selected, look in the lower half of the component browser window, specifically the Inspector tab.
- III. Find the “Drop-Down” section and add the new value into the “items” comma separated list.
- IV. The “value” listed above the items list is the default option that the drop down shows
- V. Alternatively, the 3 dots can be used to use a vision version of these operations. This window has the ability to add, subtract, or rename the values as well as select the default.



4.4 Generate Tab Options Function

- I. Copy one of the elseif statements and paste it below the previous one, but above the else statement
- II. Change the string value within the strcmp() function to represent the name of the excel file
- III. Define the “default_tab” as the string of the tab that you would like to be showed first
- IV. Change the *Name*MatrixProperties() function callback to be the name of the script created in (A.3.2)

```
562 % GenerateTabOptions - Generates the Tab options for premade Matrices
563 % - by calling upon the correct ###MatrixProperties script
564 % default_tab = allows for the tab drop down to be filled once a file is chosen
565 % tab_name = the selected tab within the dropdown
566 function GenerateTabOptions(app, file)
567
568 if strcmp(app.FileDropDown.Value, 'Bipart.xlsx')
569     default_tab = 'TAMUS21';
570     tab_name = default_tab;
571
572 [~, ~, ~, tabs] = BipartMatrixProperties(tab_name, file);
573 app.TabDropDown.Items = tabs;
574
575 elseif strcmp(app.FileDropDown.Value, 'Test.xlsx')
576     default_tab = 'Test1';
577     tab_name = default_tab;
578
579 [~, ~, ~, tabs] = TestMatrixProperties(tab_name, file);
580 app.TabDropDown.Items = tabs;
581
582 elseif strcmp(app.FileDropDown.Value, 'PowerGrid.xlsx')
583     default_tab = 'Original 5 Bus';
584     tab_name = default_tab;
585
586 [~, ~, ~, tabs] = PowerGridMatrixProperties(tab_name, file);
587 app.TabDropDown.Items = tabs;
588
589 else
590     app.TabDropDown.Items = {'N/A'};
591
592 end
593 end
```

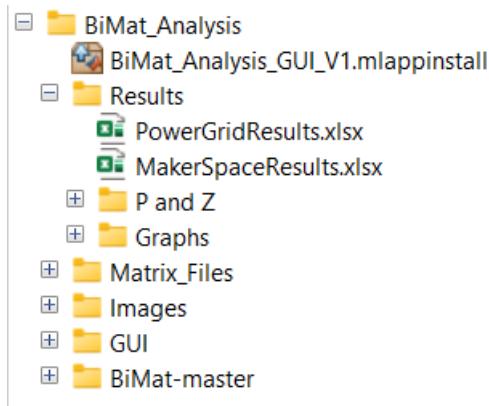
4.5 Construct Premade Matrix Function

- I. Copy one of the elseif statements and paste it below the previous one, but above the end statement
- II. Change the string value within the strcmp() function to represent the name of the excel file
- III. Change the *Name*MatrixProperties() function callback to be the name of the script created in (A.3.2)

```
596 % ConstructPremadeMatrix - Fills the Matrix ranges using the Pre-set ranges within the sheets persepective matlab script
597 %           - by calling upon the correct ###MatrixProperties script
598 function [matrix_range, row_labels_range, col_labels_range] = ConstructPremadeMatrix(app)
599
600 [~,file] = FilePath(app);
601 tab_name = app.TabDropDown.Value;
602
603 if strcmp(app.FileDropDown.Value, "Bipart.xlsx")
604     app.TabsCheckBox.Value = 1;
605     app.TabEditField.Visible = "on";
606     [matrix_range, row_labels_range, col_labels_range, ~] = BipartMatrixProperties(tab_name, file);
607 elseif strcmp(app.FileDropDown.Value, "Test.xlsx")
608     [matrix_range, row_labels_range, col_labels_range, ~] = TestMatrixProperties(tab_name, file);
609 elseif strcmp(app.FileDropDown.Value, "PowerGrid.xlsx")
610     [matrix_range, row_labels_range, col_labels_range, ~] = PowerGridMatrixProperties(tab_name, file);
611 end
612
613 end
```

5 Adding a Results Sheet

It is not necessary to add a results sheet to the program files, because the export function can create a new sheet if the specified sheet does not exist yet. However, if the user would like to do so, they can simply save an excel sheet to the results folder within the “BiMat_Analysis” folder. It is also possible for the location value to be set when the defaults are assigned. (A.6.2)

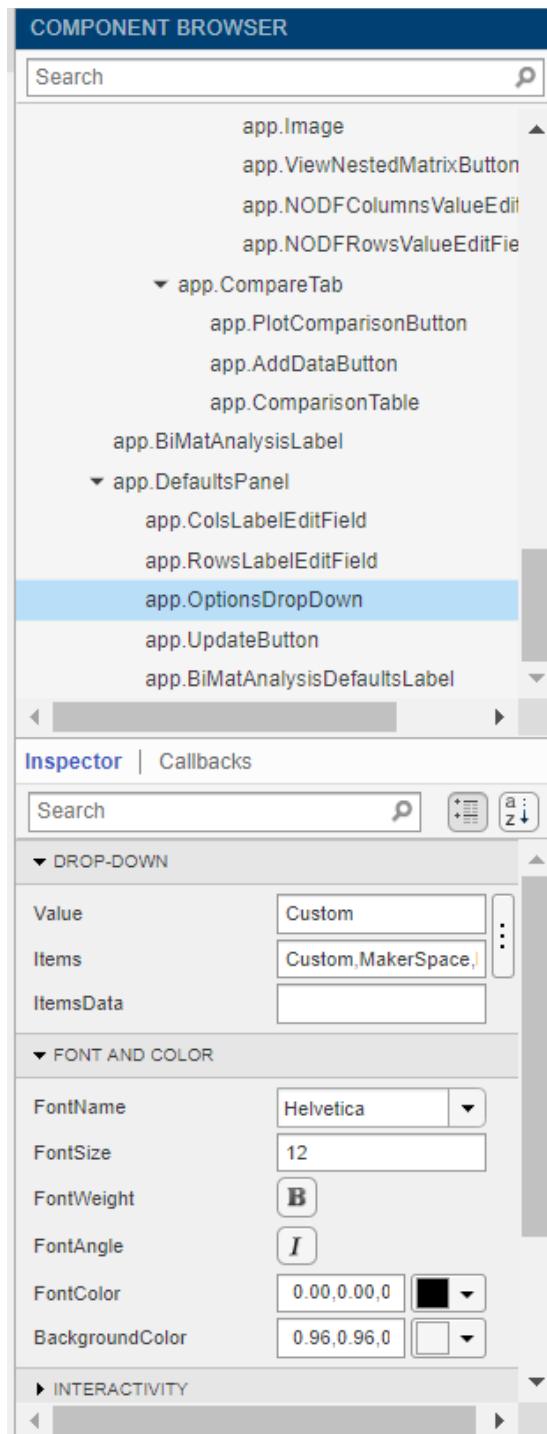


6 Adding a Default

6.1 Adding the Dropdown Value

- I. Use the Component designer within AppDesigner to locate and select the “app.OptionsDropDown”.
- II. Once selected, look in the lower half of the component browser window, specifically the Inspector tab.

- III. Find the “Drop-Down” section and add the new value into the “items” comma separated list.
- IV. The value listed above the items list is the default option that the drop down shows
- V. Alternatively, the 3 dots can be used to use a vision version of these operations. This window has the ability to add, subtract, or rename the values as well as select the default.

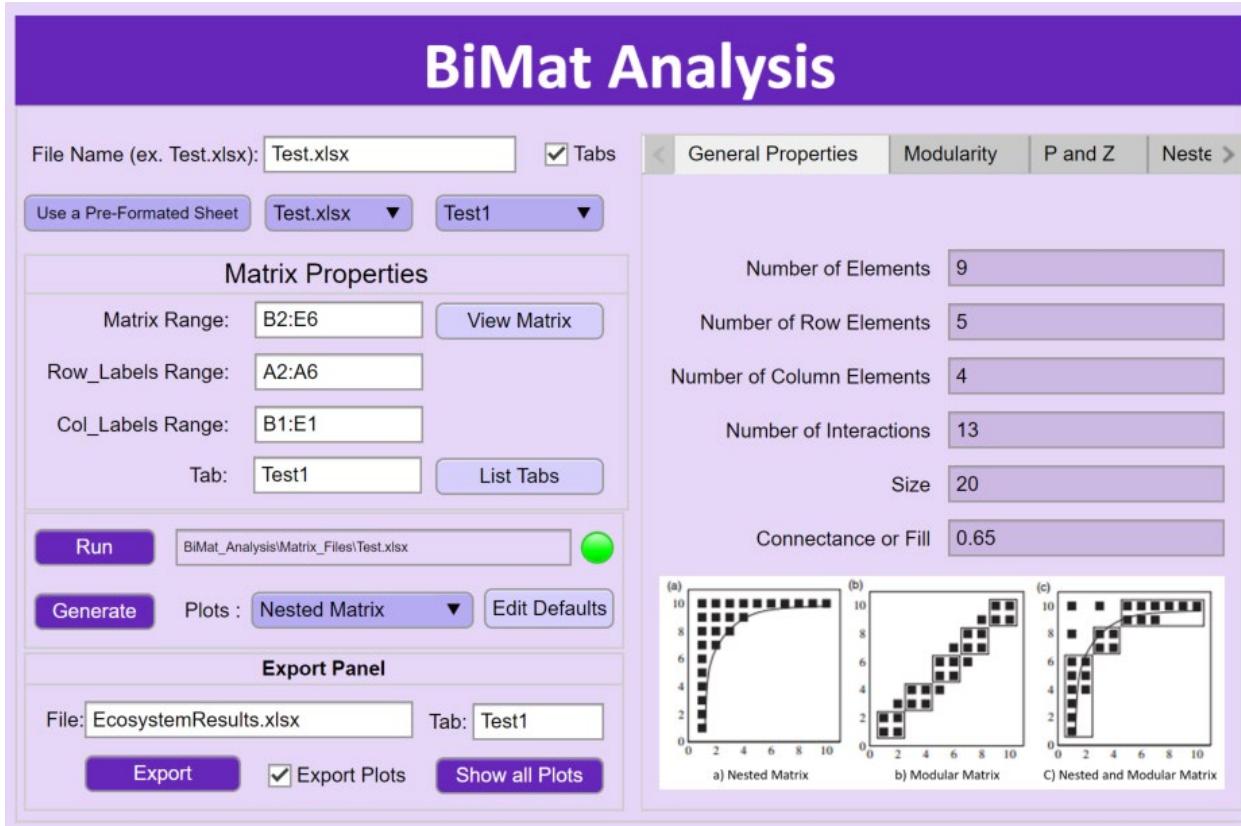


6.2 Editing the Options Drop Down Value Changed callback(B.1.3.2)

- I. Locate the “OptionDropDownValueChanged” callback (B.1.3.2)
- II. Copy and paste one of the elseif statements
- III. Set the Row and Column Labels by editing their corresponding edit field values
- IV. It is also possible to set the Results file name within this elseif statement

```
969 % Value changed function: OptionsDropDown
970 function OptionsDropDownValuechanged(app, event)
971     value = app.OptionsDropDown.Value;
972
973     app.Lamp.Color = 'red';
974
975     % sets the row and col labels as well as the file name for the results based on the default chosen
976     if strcmp(value, 'MakerSpace')
977         app.RowsLabelEditField.Value = 'Students';
978         app.ColsLabelEditField.Value = 'Tools';
979         app.FileEditField.Value = 'MakerSpaceResults.xlsx';
980     elseif strcmp(value, 'Ecosystem')
981         app.RowsLabelEditField.Value = 'Animals';
982         app.ColsLabelEditField.Value = 'Plants';
983         app.FileEditField.Value = 'EcosystemResults.xlsx';
984     elseif strcmp(value, 'Social Network')
985         app.RowsLabelEditField.Value = 'People';
986         app.ColsLabelEditField.Value = 'Events';
987     elseif strcmp(value, 'Transportation Network')
988         app.RowsLabelEditField.Value = 'Locations';
989         app.ColsLabelEditField.Value = 'Routes';
990     elseif strcmp(value, 'Power Grid')
991         app.RowsLabelEditField.Value = 'Suppliers';
992         app.ColsLabelEditField.Value = 'Receivers';
993         app.FileEditField.Value = 'PowerGridResults.xlsx';
994     end
995 end
```

B: Program Description



1 Main Page

1.1 Analysis Functions

1.1.1 File Path

```

115 % FilePath - Defines the path for file within the "Matrix_Files" folder
116 % - by combining the preset path with the manually inputted file name
117 % path = the address for the file of choice excluding the actual file name
118 % - this is currently locked to the Matrix Files folder within the master file
119 % file = the actual file name for the file
120 % - this is a manual input within the GUI
121 function [path, file] = FilePath(app)
122     file = app.FileNameexTestxlsxEditField.Value;
123     origin = "BiMat_Analysis\Matrix_Files\"; % this what forces the file location to be in the Matix_Files folder
124     path = append(origin, file);
125 end

```

1.1.1.1 Purpose

Defines the location or path for the program to use when scrapping the data from the file. It does so by combining the path to the Matrix Files folder with the file name. The file location must be within the “Matrix_Files” Folder

- The file location must be within the “Matrix_Files” Folder

1.1.1.2 Variables

- Output Variables
 - file – the filename for the sheet that is being used
 - Specifically, the File Name Edit Field at the top left corner

- path – the combination of the Origin and File
- Operation Variables
 - origin – the premade path to the “Matrix_Files” folder

1.1.2 Define Matrix

```

128 % DefineMatrix - Defines the Matrix using the ranges from the Matrix Properties panel
129 % - by reading the selected file and using the ranges to find the correct data
130 %
131 % matrix_range = the cells that contain the actual data for the Adjacency Matrix
132 % - this can be defined manually or by using a predefined sheet
133 % row_labels_range = the cells that contain the row labels for the Adjacency Matrix
134 % - this can be defined manually or by using a predefined sheet
135 % col_labels_range = the cells that contain the column labels for the Adjacency Matrix
136 % - this can be defined manually or by using a predefined sheet
137 % tab = the tab within the specified sheet that the function will be using
138 % matrix = the binary matrix that represents the Adjacency Matrix Data moving forward
139 % row_labels = the matrix that represents the Adjacency Matrix row labels moving forward
140 % col_labels = the matrix that represents the Adjacency Matrix column labels moving forward
141
142 [path,file] = FilePath(app);
143 matrix_range = app.MatrixRangeEditField.Value;
144 row_labels_range = app.Row_LabelsRangeEditField.Value;
145 col_labels_range = app.Col_LabelsRangeEditField.Value;
146
147 % Defining the Matrix depending on if the sheet has tabs or not
148 if app.TabsCheckBox.Value == 1
149   tab = app.TabEditField.Value;
150   matrix = readmatrix(path,'Sheet',tab, 'Range', matrix_range);
151   row_labels = readmatrix(path,'Sheet',tab,'Range',row_labels_range,'OutputType','string');
152   col_labels = readmatrix(path,'Sheet',tab,'Range',col_labels_range,'OutputType','string');
153 else
154   raw_tabs = sheetnames(file);
155   tab = raw_tabs(1,1);
156   matrix = readmatrix(path,'Sheet',tab, 'Range', matrix_range);
157   row_labels = readmatrix(path,'Sheet',tab,'Range',row_labels_range,'OutputType','string');
158   col_labels = readmatrix(path,'Sheet',tab,'Range',col_labels_range,'OutputType','string');
159 end
160
161 matrix = matrix > 0;
162 matrix = double(matrix);
163
164 app.StatusBar.Value = path; %shows the entire file location in the status bar
165
166 end

```

1.1.2.1 Purpose

Defines the Matrix using the ranges from the Matrix Properties panel and then reading the selected file. It does so by using the ranges to find the correct cells within the specified sheet and stores the data within them as an array. These ranges are split into three groups: Row Labels, Column Labels, and the Adjacency Matrix.

- The user may or may not be using tabs within their sheet, which is what the Tabs Check box(1.3.10) is for.
 - The Define Matrix function checks if this box is in fact checked or not to define a location within the sheet of interest.
 - If it is not checked, the function will automatically select the first tab no matter what it is named.
- All of the Range Values, as well as the Tab value, can be defined manually or by using a preformatted matrix.
 - Either way this function uses the values that are within the edit fields on the matrix properties panel. Which is why the preformatted button is necessary (1.3.5). This button will fill the

edit fields based on the values chosen in the file and tab drop downs.

1.1.2.2 Variables

- Output Variables
 - matrix - the matrix that represents the Adjacency Matrix moving forward
 - row_labels - the matrix the represents the Adjacency Matrix row labels moving forward
 - col_labels - the matrix the represents the Adjacency Matrix column labels moving forward
- Operation Variables
 - matrix_range - the cells that contain the actual data for the Adjacency Matrix
 - row_labels_range - the cells that contain the row labels for the Adjacency Matrix
 - col_labels_range - the cells that contain the column labels for the Adjacency Matrix
 - raw_tabs – is an array of all the tab names within the specified sheet
 - raw_tabs is only used when the Tabs Check Box(1.3.10) is not checked
 - tab - the tab within the specified sheet that the function will be using

1.1.3 Define Bipartite Matrix

```
169 % DefineBipartiteMatrix - Defines the Bipartite Matrix
170 % - by using the Bipartite function within the main folder of BiMat Analysis
171 %
172 % bp = the BipartiteMatrix defined by BiMat_Analysis' primary script
173 % bp_row_labels = the row labels of the bp matrix
174 % bp_col_labels = the col labels of the bp matrix
175 function [bp, bp_row_labels, bp_col_labels] = DefineBipartiteMatrix(app)
176     [matrix, row_labels, col_labels] = DefineMatrix(app);
177     bp_row_labels = row_labels;
178     bp_col_labels = col_labels;
179
180     bp = Bipartite(matrix);
181     bp.row_labels = bp_row_labels;
182     bp.col_labels = bp_col_labels;
183
end
```

1.1.3.1 Purpose

Defines the Bipartite Matrix by using the Bipartite function in the Bipartite.m script within the main folder of BiMat Analysis. The function Bipartite creates a bipartite object using a quantitative matrix web.

- See the Bipartite.m script within the “main” folder of BiMat-master.
- The bp_row and bp_col labels are equivalent to the adjacency matrix labels

1.1.3.2 Variables

- Output Variables
 - bp_row_labels – the row labels that will be used later for the bp obj
 - bp_col_labels - the col labels that will be used later for the bp obj
 - bp – the bipartite obj defined by the Bipartite function within the Bipartite.m script
 - bp.row_labels – the row labels for the bp object
 - bp.col_labels – the col labels for the bp object
- Operation Variables
 - matrix - the matrix defined by the Define Matrix Function (1.1.2)

1.1.4 General Properties

```
186 % General Properties - Defines and Displays the General Properties of the Matrix
187 % - by either calculating the values or calling on functions within MatrixFunctions.m
188 %
189 % webmatrix = the same as "matrix" but not necessarily binary
190 % num_of_species = the combined amount of both the row and column nodes in the Bipartite Matrix
191 % num_of_row_species = the amount rows in the Bipartite Matrix
192 % num_of_col_species = the amount Cols in the Bipartite Matrix
193 % num_of_interactions = the amount of rows and columns that interact(the number of cells that contain a "1")
194 % m_size = the amount of possible interactions (the amount of cells within the Bipartite Matrix)
195 % connectance = the Fill of "webmatrix"
196 function [num_of_species, num_of_row_species, num_of_col_species, num_of_interactions, m_size, connectance] = GeneralProperties(app)
197 [matrix, ~, ~] = DefineMatrix(app);
198 webmatrix = 1 * (matrix > 0);
199
200 % Calculating Values
201 num_of_species = size(matrix,1) + size(matrix,2);
202 [num_of_row_species,num_of_col_species] = size(matrix);
203 num_of_interactions = sum(webmatrix,"all");
204 m_size = num_of_col_species * num_of_row_species;
205 connectance = num_of_interactions/m_size;
206
207 % Displaying Values
208 app.NumberOfElementsEditField.Value = num_of_species;
209 app.NumberOfRowElementsEditField.Value = num_of_row_species;
210 app.NumberOfColumnElementsEditField.Value = num_of_col_species;
211 app.NumberOfInteractionsEditField.Value = num_of_interactions;
212 app.SizeEditField.Value = m_size;
213 app.ConnectanceOrFillEditField.Value = connectance;
214 end
```

1.1.4.1 Purpose

Defines and Displays the General Properties of the Matrix. All of these General Properties values are simple enough to be calculated within the function.

- The webmatrix is used in many of these calculations because there is no way for these calculations to consider weighted interactions.
 - All interactions must be represented by a 1, which is what the webmatrix does to the original adjacency matrix

1.1.4.2 Variables

- Output Variables
 - num_of_species – the total number of row and column species
 - num_of_row_species – the number of row species
 - num_of_col_species – the number of column species
 - num_of_interactions – the sum of all interactions within the webmatrix
 - m_size – the total number of possible interactions
 - connectance – the ratio of realized connections to possible connections in the webmatrix

- Operation Variables
 - matrix – the matrix defined by the Define Matrix Function (1.1.2)
 - webmatrix – the binary matrix that represents the Adjacency Matrix Data. Specifically, only with values of 0 and 1.

1.1.5 Modularity

```

217 % Modularity - Defines and Displays the Values for Modularity from the chosen algorithm
218 % - by calling upon the Detect function within the community folder of
219 % BiMat_Analysis once the specified algorithm is ran
220 % community = represents the Bipartite Adjacency Matrix that will be used to
221 % calcualte modularity
222 % obj = a variable frame that allows for the Modularity values to be defined
223 % based on the correct algorithm
224 % N = number of modules
225 % Qb = standard modularity metric
226 % Qr = ratio of interactions (internal/external)
227 % A = matrix that represents the row interactions
228 % B = matrix that represents the column interactions
229 function [N, Qb, Qr, A, B] = Modularity(app)
230
231 [bp, ~, ~] = DefineBipartiteMatrix(app);
232
233 % Selecting Algorithm with the Drop Down
234
235 % LeadingEigenVector
236 if strcmp(app.AlgorithmDropDown.Value, 'LeadingEigenVector') == true
237
238 % KernighanLinTunning checkbox
239 %if app.KernighanLinTunningCheckBox.Value == 0
240 %  bp.community.DoKernighanLinTunning = false;
241 %else
242 %  bp.community.DoKernighanLinTunning = true;
243 %end
244
245 bp.community = LeadingEigenvector(bp.matrix);
246
247 % ApaditiveBrim
248 elseif strcmp(app.AlgorithmDropDown.Value, 'AdaptiveBrim') == true
249   bp.community = AdaptiveBrim(bp.matrix);
250
251 % LPBrim
252 elseif strcmp(app.AlgorithmDropDown.Value, 'LPBrim') == true
253   bp.community = LPBrim(bp.matrix);
254 end
255
256 % Defining Modularity Values
257 obj = Detect(bp.community);
258 N = obj.N;
259 Qb = obj.Qb;
260 Qr = obj.Qr;
261
262 A = bp.community.row_modules;
263 B = bp.community.col_modules;
264
265 % Displaying Modularity Values
266 app.NNumberOfModulesEditField.Value = N;
267 app.QbStandardModularityMetricEditField.Value = Qb;
268 app.QrRatioofintertextEditField.Value = Qr;
269
270 end

```

1.1.5.1 Purpose

Defines and Displays the Values for Modularity from the chosen algorithm (1.3.14). It does so by calling upon the Detect function within the “community” folder of BiMat_Master once the specified algorithm is ran.

- See the LeadingEigenVector.m, AdaptiveBrim.m, LPBrim.m, and BipartiteModularity.m scripts within the community folder in BiMat-Master.

- Creates a *selected algorithm* (1.3.14) object “obj” using a bipartite matrix that will then be used to calculate modularity
- Uses the Detect function within BipartiteModularity.m to calculate the modularity by first dividing the network (matrix) into isolated components. The modularity is then optimized in each component separately. The modularity is optimized globally by default, for optimizing components independently of the rest change properties “optimize_by_component” to true before calling this method.

1.1.5.2 Variables

- Output Variables
 - N – the number of modules
 - Qb – Standard Modularity function
 - Qr – Ratio of interactions (internal/external)
 - A – matrix that represents the row interactions
 - B – matrix that represents the column interactions
- Operation Variables
 - bp – the bipartite obj defined by the Define Bipartite Matrix Function (1.1.3)
 - bp.community – represents the Bipartite Adjacency Matrix that will be used to calculate modularity
 - will be run in any of the three algorithm scripts: LeadingEigenVector.m, AdaptiveBrim.m, or LPBrim.m depending on the value of the Algorithm Dropdown (1.3.14)
 - obj – a variable frame that allows for the Modularity values to be defined using the Detect function in the community folder, and is based on the correct algorithm

1.1.6 Nestedness

```

273 % Nestedness - Defines and Displays the Values for Nestedness
274 % - by calling upon the Detect function within the nestedness.m script
275 % within the nestedness folder of BiMat_Analysis
276 % obj = a variable frame that allows for the Nestedness values to be defined
277 % nestedness = the nestedness of the Bipartite matrix
278 % N_rows =
279 % N_cols =
280 function [nestedness, N_rows, N_cols] = Nestedness(app)
281
282 % [bp, ~, ~] = DefineBipartiteMatrix(app);
283
284 % Defining Nestedness Values
285 % obj = Detect(bp.nestedness);
286 % nestedness = obj.N;
287 % N_rows = obj.N_rows;
288 % N_cols = obj.N_cols;
289
290 % Displaying Nestedness Value
291 % app.NODFNestednessValueEditField.Value = nestedness;
292 % app.NODFRowsValueEditField.Value = N_rows;
293 % app.NODFColumnsValueEditField.Value = N_cols;
294 end

```

1.1.6.1 Purpose

Defines and Displays the Values for Nestedness by calling upon the Detect function within the nestedness.m script within the nestedness folder of BiMat_Analysis.Nestedness.m uses the nestednessNODF.m to define the actual values for nestedness.

- See nestedness.m and nestednessNODF.m scripts within the nestedness folder in BiMat-Master

1.1.6.2 Variables

- Output Variables
 - nestedness – the combined NODF nestedness value
 - N_rows – the NODF nestedness for the rows
 - N_cols – the NODF nestedness for the cols
- Operation Variables
 - bp – the bipartite obj defined by the Define Bipartite Matrix Function (1.1.3)
 - obj – a variable frame that allows for the Nestedness values to be defined from the calculated values within the Nestedness.m script which uses the NestednessNODF.m script

1.1.7 NestedMatrix

1.1.7.1 Purpose

Displays the Matrix that the nestedness calculations use. By calling upon the “SORT_MATRIX” function within the MatrixFunctions script and showing it on the Matrix Page (B.5). Although this function sorts the matrix itself it does not sort the labels, so this function must resort them.

1.1.7.2 Variables

- Output
 - sorted_row_labels – the sorted version of the bp_row_labels
 - sorted_col_labels – the sorted version of the bp_col_labels
 - sorted_matrix – the sorted Bipartite matrix that is in the most nested form defined by the the “SORT_MATRIX” function within the MatrixFunctions script.
- Operational Variables
 - bp – the bipartite obj defined by the Define Bipartite Matrix Function (B.1.1.3)
 - bp_row_labels – the row labels defined by the Define Bipartite Matrix function (B.1.1.3)

- bp_col_labels - the column labels defined by the Define Bipartite Matrix function (B.1.1.3)
- ir – an array of the row's (index_row) indices positioned in their new positions. Not the actual values but the original index number reordered. Defined by the the “SORT_MATRIX” function within the MatrixFunctions script.
- ⊖ ic – an array of the col's (index_col) indices positioned in their new positions. Not the actual values but the original index number reordered. Defined by the the “SORT_MATRIX” function within the MatrixFunctions script.
- col_index – a dummy variable defined by the specific column index from the ic array
- row index – a dummy variable defined by the specific row index from the ir array

1.1.8 P and Z

```

297 % PandZ - Solves for the P and z values based on the Row or Col Modules
298 % - by
299 % C = the maximum value within the "A" matrix
300 % D = the maximum value within the "B" matrix
301 % Nm = the maximum value between "C" and "D"
302 % Nsm = an empty array with the ordered values of Nc and Nr
303 % Nr = the length of the row matrix ("A")
304 % Nc = the length of the row matrix ("B")
305 % kir = the Total Degree or the row nodes
306 % kic = the Total Degree or the column nodes
307 % kitr = the links from row node i to a specific module
308 % kitc = the links from column node i to a specific module
309 % P = a operational variable that allows for Pr or Pc to be defined
310 % Pr = the P values for the row nodes
311 % Pc = the P values for the column nodes
312 % kisr = the links from row node i to other nodes within its own module
313 % kisc = the links from column node i to other nodes within its own module
314 % kim = an array of each kis value for each module (for mean and stdev)
315 % zr = the z values for the row nodes
316 % zc = the z values for the col nodes
317 function [Pr, zr, Pc, zc] = PandZ(app)

318 %[matrix, row_labels, col_labels] = DefineMatrix(app);
319 %[bp, bp_row_labels, bp_col_labels] = DefineBipartiteMatrix(app);
320 matrix = bp.matrix;

321 %~, ~, ~, A, B] = Modularity(app);

322 % Finding participation (connectivity)(with other modules)
323 % Finding ki values for all row and column species
324 %Nr=length(A);
325 %D=max(B);
326 %Nm=max(C,D);

327 % Finding participation (connectivity)(with other modules)
328 % Finding ki values for all row and column species
329 %Nr=length(A);
330 %Nc=length(B);
331 %Kir=zeros(Nr,1);
332 %for i=1:Nr
333 %    Kir(i)=sum(matrix(i,:));
334 %end
335 %Kic=zeros(Nc,1);
336 %for i=1:Nc
337 %    Kic(i)=sum(matrix(:,i));
338 %end
339 %
340 %

```

```

341 % Finding kit values for all row and column species
342 kitr=zeros(Nr,Nm);
343 kitc=zeros(Nc,Nm);
344 for i=1:Nr
345     for j=1:Nc
346         if matrix(i,j)==1
347             mod=B(j);
348             kitr(i,mod)=kitr(i,mod)+1;
349         end
350     end
351 end
352 for j=1:Nc
353     for i=1:Nr
354         if matrix(i,j)==1
355             mod=A(i);
356             kitc(j,mod)=kitc(j,mod)+1;
357         end
358     end
359 end
360
361 % Calculating P for each species
362 Pr=zeros(Nr,1);
363 Pc=zeros(Nc,1);
364 for i=1:Nr
365     P=1;
366     for j=1:Nm
367         P=P-(kitr(i,j)/kir(i))^2;
368     end
369     Pr(i)=P;
370 end
371 for i=1:Nc
372     P=1;
373     for j=1:Nm
374         P=P-(kitc(i,j)/kic(i))^2;
375     end
376     Pc(i)=P;
377 end
378
379 % Finding intra-module degree (z) (within their own module)
380 % Finding number of species in each module
381 Nsm=zeros(Nm,1);
382 for i=1:Nm
383     for j=1:Nr
384         if A(j)==i
385             Nsm(i)=Nsm(i)+1;
386         end
387     end
388     for j=1:Nc
389         if B(j)==i
390             Nsm(i)=Nsm(i)+1;
391         end
392     end
393 end
394

```

```

395 % Finding kis values for all row and column species
396 kisr=zeros(Nr,1);
397 kisc=zeros(Nc,1);
398 for i=1:Nr
399     kisr(i)=kitr(i,A(i));
400 end
401 for i=1:Nc
402     kisc(i)=kitc(i,B(i));
403 end
404
405
406 % Creating matrix of each kis value for each module (for mean and stdev)
407 kim=zeros(Nm,max(Nsm));
408 kim=ones(Nm,1); % This will be used to insert each new species into a blank space in the appropriate row
409 for i=1:Nr
410     mod=A(i);
411     kim(mod,kimIndex(mod))=kisr(i);
412     kimIndex(mod)=kimIndex(mod)+1;
413 end
414 for i=1:Nc
415     mod=B(i);
416     kim(mod,kimIndex(mod))=kisc(i);
417     kimIndex(mod)=kimIndex(mod)+1;
418 end
419
420 % Finding mean and stdev values for each module's kis values
421 meanstdev=zeros(Nm,2);
422 for i=1:Nm
423     meanstdev(i,1)=mean(kim(i,1:Nsm(i)));
424     meanstdev(i,2)=std(kim(i,1:Nsm(i)));
425 end
426
427 % Calculating z for each species
428 zr=zeros(Nr,1);
429 zc=zeros(Nc,1);
430 for i=1:Nr
431     zr(i)=(kisr(i)-meanstdev(A(i),1))/(meanstdev(A(i),2));
432 end
433 for i=1:Nc
434     zc(i)=(kisc(i)-meanstdev(B(i),1))/(meanstdev(B(i),2));
435 end
436
437 % Compiling final tables
438 ROutput=[bp_row_labels',Pr,zr];
439 COutput=[bp_col_labels',Pc,zc];
440
441 end

```

1.1.8.1 Purpose

Solves for the P and Z values of each node based on both the row and column elements. The method used to obtain these values can only work on one group at a time, which is why there is a distinction between rows and columns. This function iteratively defines them both semi-simultaneously and outputs them as separate arrays for both P and z.(Pr, Pc, zr, zc)

1.1.8.2 Variables

- Output Variables
 - Pr – an array of P values for the row elements
 - zr – an array of z values for the row elements
 - Pc – an array of P values for the column elements
 - zc – an array of z values for the column elements
- Operation Variables
 - bp – the bipartite obj defined by the Define Bipartite Matrix Function (1.1.3)
 - bp_row_labels – the row labels defined by the Define Bipartite Matrix function (1.1.3)
 - bp_col_labels - the column labels defined by the Define Bipartite Matrix function (1.1.3)

- matrix – the adjacency matrix that used in the analysis defined by the bipartite object, bp.
- A – an array that represents the row interactions defined by the Modularity function (B.1.1.5)
- B – an array that represents the col interactions defined by the Modularity function (B.1.1.5)
- C – the maximum value within the A array
- D – the maximum value within the B array
- Nm – the maximum value between the C and D array
- Nc – the length of the col matrix (“A”)
- Nr – the length of the row matrix (“B”)
- kir – the total degree of the row nodes
- kic – the total degree of the col nodes
- kitr – the links from row node i to a specific module
- kitc – the links from column node i to a specific module
- Nsm – an empty array with the ordered values of Nc and Nr
- kisr – the links from row node i to other nodes within its own module
- kisc – the links from column node i to other nodes within its own module
- kim – an array of each kis value for each module (for mean and stdev)
- kimIndex – an array used to insert each new species into a blank space in the appropriate row
- meanstdev – an array of the mean and stdev values for each module's kis values

1.1.9 Define Roles

```

444 % DefineRoles - Assigns each node to their role based on their P and Z values
445 % by iterating through each node's correlated P and Z values
446 %
447 % N = the number of nodes within the matrix
448 % ultra_peripheral = a reordered column array of the nodes that are defined by the ultra peripheral role
449 % peripheral = a reordered column array of the nodes that are defined by the peripheral role
450 % nonhub_connector = a reordered column array of the nodes that are defined by the non-hub connector role
451 % nonhub_kinless = a reordered column array of the nodes that are defined by the non-hub kinless role
452 % provincial = a reordered column array of the nodes that are defined by the provincial role
453 % connector = a reordered column array of the nodes that are defined by the connector role
454 % kinless = a reordered column array of the nodes that are defined by the kinless role
455 % roles = a combined array containing all of the role arrays above
456 % role_col = a combined column array of roles that remains in the original order
457 function [roles, ultra_peripheral, peripheral, nonhub_connector, nonhub_kinless, provincial, connector, kinless, role_col] = DefineRoles(mpp, p, z, labels)
458 N = length(p); %Gets the length of the iteration
459
460 % Initializes empty role arrays
461 ultra_peripheral = [];
462 peripheral = [];
463 nonhub_connector = [];
464 nonhub_kinless = [];
465 provincial = [];
466 connector = [];
467 kinless = [];
468 roles = strings([N,7]);
469 role_col = strings([N,1]);
470
471 % Fills the role arrays
472 for i=1:N
473 % Non-Hub Roles
474 if z(i) < 2.5
475 if p(i) <=.1
476 ultra_peripheral = [ultra_peripheral; labels(i)];
477 role_col(i, 1) = 'Ultra-Peripheral';
478 elseif p(i) >=.1 && p(i) <=.625
479 peripheral = [peripheral; labels(i)];
480 role_col(i, 1) = 'Non-Hub Peripheral';
481 elseif p(i) >=.625 && p(i) <=.8
482 nonhub_connector = [nonhub_connector; labels(i)];
483 role_col(i, 1) = 'Non-Hub Connector';
484 elseif p(i) >=.8
485 nonhub_kinless = [nonhub_kinless; labels(i)];
486 role_col(i, 1) = 'Non-Hub Kinless';
487 end
488 % Hub Roles
489 elseif z(i) >= 2.5
490 if p(i) <=.3
491 provincial = [provincial; labels(i)];
492 role_col(i, 1) = 'Provincial';
493 elseif p(i) >=.3 && p(i) <=.75
494 connector = [connector; labels(i)];
495 role_col(i, 1) = 'Connector';
496 elseif p(i) >=.75
497 kinless = [kinless; labels(i)];
498 role_col(i, 1) = 'Non-Hub Kinless';
499 end
500 end
501
502 % Reordering the Nodes
503 for i=1:length(ultra_peripheral)
504 roles(i,1) = ultra_peripheral(i);
505 end
506 for i=1:length(peripheral)
507 roles(i,2) = peripheral(i);
508 end
509 for i=1:length(nonhub_connector)
510 roles(i,3) = nonhub_connector(i);
511 end
512 for i=1:length(nonhub_kinless)
513 roles(i,4) = nonhub_kinless(i);
514 end
515 for i=1:length(provincial)
516 roles(i,5) = provincial(i);
517 end
518 for i=1:length(connector)
519 roles(i,6) = connector(i);
520 end
521 for i=1:length(kinless)
522 roles(i,7) = kinless(i);
523 end
524
525 end

```

1.1.9.1 Purpose

Assigns each element to a role based on its P and Z values. Similar to the P and Z function, this can only do one at a time. Therefore, either the row elements or the column elements can be defined.

However, unlike the P and Z function the Define Roles function can't define them semi-simultaneously. This is why the input variables are simply P, z, and labels rather than Pr, zr, and row_labels.

- The ModulesSwitch(1.3.16) can be used to toggle between row or column elements in the analysis.

1.1.9.2 Variables

- Output Variables

- ultra_peripheral – a column array of the nodes that are ultra-peripherals
- peripheral – a column array of the nodes that are peripherals
- nonhub_connector – a column array of the nodes that are nonhub connectors
- nonhub_kinless – a column array of the nodes that are nonhub kinless'
- provincial – a column array of the nodes that are provincials
- connector – a column array of the nodes that are connectors
- kinless – a column array of the nodes that are kinless'
- roles – a combined array of all the role column arrays
- role_col – a single column of the roles that are assigned to each element in their original order
- Input Variables
 - P – an array of the P values for each node
 - z – an array of the z values for each node
 - labels – an array of the names for each node
- Operation Variables
 - N – the length of iteration that the role defining steps use.
Defined by the length of the P array.

1.2 GUI Functions

1.2.1 Run BiMat

```

527 % RunBiMat - The Main function of the GUI that steps though the analysis functions
528 function RunBiMat(app)
529
530 % General Properties - Defines and Displays the General Properties of the Matrix
531 GeneralProperties(app);
532
533 % Modularity - Defines and Displays the Values for Modularity from the chosen algorithum
534 Modularity(app);
535
536 % Nestedness - Defines and Displays the Values for Nestedness
537 Nestedness(app);
538
539 % FillPandZ - Combines the P and Z values with their roles and Displays them
540 FillPandZ(app);
541
542 % Progress Lamp
543 app.Lamp.Color = '0.00,1.00,0.00'; % Green
544
545 end

```

1.2.1.1 Purpose

The Main function of the GUI that steps though the analysis functions.

1. General Properties() (1.1.4)
2. Modularity() (1.1.5)
3. Nestedness() (1.1.6)

4. FillPandZ() (1.1.8)

1.2.2 Generate Plots

```
548 % GeneratePlots - Shows the Plot that is Chosen in the Dropdown as a seperate window
549 function GeneratePlots(app)
550
551 [bp, ~, ~] = DefineBipartiteMatrix(app);
552 f = figure('Visible','on');
553
554 if strcmp(app.PlotsDropDown.Value, 'Modular Graph')
555 bp.plotter.PlotModularGraph();
556 elseif strcmp(app.PLOTSDROPDOWN.VALUE, 'Matrix')
557 bp.plotter.PlotMatrix();
558 elseif strcmp(app.PLOTSDROPDOWN.VALUE, 'Modular Matrix')
559 bp.plotter.PlotModularMatrix();
560 elseif strcmp(app.PLOTSDROPDOWN.VALUE, 'Nested Matrix')
561 bp.plotter.PlotNestedMatrix();
562 end
563
564 end
```

1.2.2.1 Purpose

Generates the Plot that is Chosen in the Plot Dropdown as a separate window. It does so by using the bp object with the specific plotting functions within the PlotWebs.m script within the “draw” folder of BiMat-Master

1.2.2.2 Variables

- bp – the bipartite obj defined by the Define Bipartite Matrix Function(1.1.3)

1.2.3 Generate Tab Options

```
567 % GenerateTabOptions - Generates the Tab options for premade Matrices
568 % - by calling upon the correct ##MatrixProperties script
569 % default_tab = allows for the tab drop down to be filled once a file is chosen
570 % tab_name = the selected tab within the dropdown
571 function GenerateTabOptions(app, file)
572
573 if strcmp(app.FileDropDown.Value, 'Bipart.xlsx')
574 default_tab = 'TAMUS21';
575 tab_name = default_tab;
576
577 [~, ~, tabs] = BipartMatrixProperties(tab_name, file);
578 app.TabDropDown.Items = tabs;
579
580 elseif strcmp(app.FileDropDown.Value, 'Test.xlsx')
581 default_tab = 'Test1';
582 tab_name = default_tab;
583
584 [~, ~, tabs] = TestMatrixProperties(tab_name, file);
585 app.TabDropDown.Items = tabs;
586
587 elseif strcmp(app.FileDropDown.Value, 'PowerGrid.xlsx')
588 default_tab = 'Original 5 Bus';
589 tab_name = default_tab;
590
591 [~, ~, tabs] = PowerGridMatrixProperties(tab_name, file);
592 app.TabDropDown.Items = tabs;
593
594 else
595 app.TabDropDown.Items = {'N/A'};
596
597 end
598 end
```

1.2.3.1 Purpose

Generates the Tab options (all the tabs within the specified sheet) for premade Matrices by calling upon the correct “*Name*MatrixProperties.m” script.

1.2.3.2 Variables

- Input Variables
 - file - the filename for the sheet that is being used defined by the File Path function (B.1.1.1)
- Operation Variables
 - tabs – an array of the different tabs within the specified sheet
 - default_tab – specified when the sheet is formatted and allows for the tab drop down to be filled once a file is chosen
 - tab_name - the selected tab within the dropdown

1.2.4 Construct Premade Matrix

```
601 % ConstructPremadeMatrix - Fills the Matrix ranges using the Pre-set ranges within the sheets perspective matlab script
602 %
603 % - by calling upon the correct ###MatrixProperties script
604 function [matrix_range, row_labels_range, col_labels_range] = ConstructPremadeMatrix(app)
605
606 [~,file] = FilePath(app);
607 tab_name = app.TabDropDown.Value;
608
609 if strcmp(app.FileDropDown.Value, "Bipart.xlsx")
610     app.TabsCheckBox.Value = 1;
611     app.TabEditField.Visible = "on";
612     [matrix_range, row_labels_range, col_labels_range, ~] = BipartMatrixProperties(tab_name, file);
613 elseif strcmp(app.FileDropDown.Value, "Test.xlsx")
614     [matrix_range, row_labels_range, col_labels_range, ~] = TestMatrixProperties(tab_name, file);
615 elseif strcmp(app.FileDropDown.Value, "PowerGrid.xlsx")
616     [matrix_range, row_labels_range, col_labels_range, ~] = PowerGridMatrixProperties(tab_name, file);
617 end
618
```

1.2.4.1 Purpose

Fills the Matrix ranges using the Pre-set ranges within the sheets perspective MATLAB script by calling upon the correct “*Name*MatrixProperties.m” script.

1.2.4.2 Variables

- Output Variables
 - matrix_range - the cells that contain the actual data for the Adjacency Matrix defined by the “*Name*MatrixProperties.m”
 - row_labels_range - the cells that contain the row labels for the Adjacency Matrix defined by the “*Name*MatrixProperties.m”
 - col_labels_range - the cells that contain the column labels for the Adjacency Matrix defined by the “*Name*MatrixProperties.m”
- Operation Variables

- file – defined by the FilePath function(1.1.1)
- tab_name – the selected tab within the dropdown

1.2.5 Fill P and Z

```

621 % FillPandZ - Combines the P and Z values with their roles and Displays them
622 % p = a column array of the P values(either based on rows or columns)
623 % z = a column array of the the z values(either based on rows or columns)
624 % labels = a column array of the the labels(either based on rows or columns)
625 % label = the labels value chosen on the default page(either based on rows or columns)
626 % - this can be defined manually or by using the predefined options
627 function FillPandZ(app)
628     [Pr, zr, Pc, zc] = PandZ(app);
629
630     [~, bp_row_labels, bp_col_labels] = DefineBipartiteMatrix(app);
631
632     % Defines Value based on either Rows or Columns
633     if strcmp(app.ModulesSwitch.Value, 'Rows')
634         p = Pr;
635         z = zr;
636         labels = bp_row_labels;
637         label = app.RowsLabelEditField.Value;
638     elseif strcmp(app.ModulesSwitch.Value, 'Columns')
639         p = Pc;
640         z = zc;
641         labels = transpose(bp_col_labels);
642         label = app.ColsLabelEditField.Value;
643     end
644
645     [~, ~, ~, ~, ~, ~, role_col] = DefineRoles(app, p, z, labels);
646
647     % Combines the P and Z scores with their node label and role.
648     app.PandZTable.Data = horzcat(labels, p, z, role_col);
649
650     % Displays the combined array
651     app.PandZTable.ColumnName = {label, 'P-Value', 'z-Value', 'Role'};
652
653 end

```

1.2.5.1 Purpose

Shows the P and Z values along with their role for each row or column element depending on the modules switch (1.3.16). This is done by combining the labels array, p array, z array, and the role_col array.

- See the Define Roles function (1.1.18) for the role assignments/definitions
- See the P and Z page for a more detailed view of the role assignments. (3)

1.2.5.2 Variables

- Operation Variables
 - Pr – an array of P values for the row elements
 - zr – an array of z values for the row elements
 - Pc – an array of P values for the column elements
 - zc – an array of z values for the column elements
 - bp_row_labels – the row labels defined by the Define Bipartite Matrix function (1.1.3)
 - bp_col_labels - the column labels defined by the Define Bipartite Matrix function (1.1.3)

- p – an array of P values either for the row or column elements based on the modules switch(1.3.16)
- z – an array of z values either for the row or column elements based on the modules switch(1.3.16)
- labels – an array of element names from either the row or column elements depending on the modules switch value(1.3.16)
- label – the last element of the P and Z pages title defined by the defaults page. It is either the row or column label in the defaults page depending on the modules switch value(1.3.16)
- role_col – a column array of the roles for each node defined by the Define Roles function (B.1.1.9).

1.2.6 Launch P and Z

```

556 % LaunchPandZ - Opens the PandZPage and plots the P and Z values
557 % on a scatter plot as well as displays their roles
558 function LaunchPandZ(app, labels, p, z, label)
559
560 [~, file] = FilePath(app);
561 tab = app.TabEditField.Value;
562 [roles, ~, ~, ~, ~, ~, ~, ~, role_col] = DefineRoles(app, p, z, labels);
563
564 % Opens the P and Z app
565 app.PandZPage = PandZPage(file, tab, labels, p, z, label, roles, role_col);
566 end

```

1.2.6.1 Purpose

Opens the P and Z Page and plots the P and Z values on a scatter plot as well as displays their roles.

- See the P and Z Page section (3) for a detailed description of what this page is used for.

1.2.6.2 Variables

- Input Variables
 - p – an array of the p values
 - z – an array of the z values
 - label – the last element of the P and Z pages title
- Operation Variables
 - file – the file named defined by the File Path function (1.1.1).
 - tab – the tab name defined in the Tab Edit Field
 - labels – an array of the element labels defined as in input to the function
 - roles – an array of the elements placed into the column that represents the different roles

- role_col – a column array that lists the elements role in the order that the labels are originally in

1.2.7 Export to Excel

```

669 % ExporttoExcel - Combines all of the GUI results into a single cell array and
670 % prepares it to be exported to a specified excel sheet and tab
671 function results = ExporttoExcel(app)
672     file = append('BiMat_Analysis\Results\', app.FileEditField.Value);
673     tab = app.TabEditField_2.Value;
674
675     [~, bp_row_labels, bp_col_labels] = DefineBipartiteMatrix(app);
676     [num_of_species, num_of_row_species, num_of_col_species, num_of_interactions, m_size, connectance] = GeneralProperties(app);
677     [N, Qb, Qr, ~, ~] = Modularity(app);
678     algorithm = app.AlgorithmDropDown.Value;
679     [nestedness, N_rows, N_cols] = Nestedness(app);
680     [Pr, zr, Pc, zc] = PandZ(app);
681     [~, ~, ~, ~, ~, ~, r_role_col] = DefineRoles(app, Pr, zr, bp_row_labels);
682     [~, ~, ~, ~, ~, ~, c_role_col] = DefineRoles(app, Pc, zc, bp_col_labels);
683     r_label = app.RowsLabelEditField.Value;
684     c_label = app.ColumnsLabelEditField.Value;
685     Title = append(app.FileEditField.Value, ' - ', tab);
686
687     if length(Pr) > length(Pc)
688         n = 20 + length(Pr);
689     else
690         n = 20 + length(Pc);
691     end
692
693     results = cell(n,8);
694
695     results{1,2} = Title;
696
697     results{3,1} = 'General Properties';
698     results{4,1} = 'Number of Elements';
699     results{4,2} = num_of_species;
700     results{5,1} = 'Number of Row Elements';
701     results{5,2} = num_of_row_species;
702     results{6,1} = 'Number of Col Elements';
703     results{6,2} = num_of_col_species;
704     results{7,1} = 'Number of Interactions';
705     results{7,2} = num_of_interactions;
706     results{8,1} = 'Size';
707     results{8,2} = m_size;
708     results{9,1} = 'Connectance or Fill';
709     results{9,2} = connectance;
710
711     results{11,1} = 'Modularity';
712     results{12,1} = 'Algorithm';
713     results{12,2} = algorithm;
714     results{13,1} = 'N - Number of Modules';
715     results{13,2} = N;
716     results{14,1} = 'Qb - Standard Modularity Metric';
717     results{14,2} = Qb;
718     results{15,1} = 'Qr - Ratio of int/ext interactions';
719     results{15,2} = Qr;
720
721     results{17, 1} = 'Nestedness';
722     results{18, 1} = 'NODF Nestedness Value';
723     results{18, 2} = nestedness;
724     results{19, 1} = 'NODF Rows Value';
725     results{19, 2} = N_rows;
726     results{20, 1} = 'NODF Columns Value';
727     results{20, 2} = N_cols;
728
729     results{22,1} = r_label;
730     results{22,2} = 'P-Value';
731     results{22,3} = 'z-Value';
732     results{22,4} = 'Role';
733     for i=1:(length(Pr))
734         results{((22+i),1)} = bp_row_labels(i);
735         results{((22+i),2)} = Pr(i);
736         results{((22+i),3)} = zr(i);
737         results{((22+i),4)} = r_role_col(i);
738     end
739
740     results{22,6} = c_label;
741     results{22,7} = 'P-Value';
742     results{22,8} = 'z-Value';
743     results{22,9} = 'Role';
744     for i=1:(length(Pc))
745         results{((22+i),6)} = bp_col_labels(i);
746         results{((22+i),7)} = Pc(i);
747         results{((22+i),8)} = zc(i);
748         results{((22+i),9)} = c_role_col(i);
749     end
750
751 end
752
753 end
754
755 end

```

1.2.7.1 Purpose

Combines all of the GUI results into a single cell array and prepares it to be exported to a specified excel sheet and tab.

- The output location must be within the results folder of the BiMat_Analysis folder.

1.2.7.2 Variables

- Output Variables
 - results – a large array that combines all of the results from general properties, Modularity, Nestedness, as well as the P and Z values.
- Operation Variables
 - file – the filename for the sheet that is being used defined by the File Path function (B.1.1.1)
 - tab – the tab within the specified sheet that the function will be using
 - bp – the bipartite obj defined by the Define Bipartite Matrix Function (B.1.1.3)
 - matrix – the matrix within the bp object from the Bipartite Matrix function(B.1.1.3)
 - bp_row_labels – the row labels defined by the Define Bipartite Matrix function (B.1.1.3)
 - bp_col_labels - the column labels defined by the Define Bipartite Matrix function (B.1.1.3)
 - num_of_species – the total number of row and column species
 - num_of_row_species – the number of row species
 - num_of_col_species – the number of column species
 - num_of_interactions – the sum of all interactions within the webmatrix
 - m_size – the total number of possible interactions
 - connectance – the ratio of realized connections to possible connections in the webmatrix
 - N – the number of modules
 - Qb – Standard Modularity function
 - Qr – Ratio of interactions (internal/external)
 - algorithm – the algorithm that's used for the modularity functions
 - nestedness – the combined NODF nestedness value
 - N_rows – the NODF nestedness for the rows
 - N_cols – the NODF nestedness for the cols
 - sorted_row_labels – the sorted version of the bp_row_labels. Defined by the Nested Matrix function (B.1.1.7)
 - sorted_col_labels – the sorted version of the bp_col_labels Defined by the Nested Matrix function (B.1.1.7)

- sorted_matrix – the sorted Bipartite matrix that is in the most nested form defined by the the “SORT_MATRIX” function within the MatrixFunctions script. Defined by the Nested Matrix function (B.1.1.7)
- Pr – an array of P values for the row elements
- zr – an array of z values for the row elements
- Pc – an array of P values for the column elements
- zc – an array of z values for the column elements
- r_role_col – a column array of the roles for each row node defined by the Define Roles function (B.1.1.9).
- c_role_col – a column array of the roles for each col node defined by the Define Roles function (B.1.1.9).
- r_label – the row label value form the defaults page defined by the option dropdown value change callback (B.1.3.2)
- c_label – the col label value form the defaults page defined by the option dropdown value change callback (B.1.3.2)
- Title – the title shown at the top right of the data defineds by combining the export file name and the tab
- n – the length of iteration that the role defining steps use. Defined by the length of either the Pr or Pc array.

1.3 Callbacks

1.3.1 Startup Function

```

758 % Code that executes after component creation
759 function startupFcn(app)
760 -    file = app.FileDropDown.Value;
761 -    app.MatrixPropertiesPanel.Visible = 'off';
762 -    app.TabEditField.Visible = 'off';
763 -    app.TabEditFieldLabel.Visible = 'off';
764 -    app.ShowallPlotsButton.Visible = 'off';
765
766 -    GenerateTabOptions(app, file);
767 -    app.Lamp.Color = 'red';
768
769 -    app.BiMatPanel.Visible = 'off';
770 -    app.DefaultsPanel.Visible = 'on';
771 - end

```

1.3.1.1 Purpose

Hides the Majority of the panels that make up the GUI, and only shows the defaults panel. It also generates the tabs for the default file within the file drop down to prevent any errors.

1.3.2 Option Drop Down Value Changed

```
773 % Value changed function: OptionsDropDown
774 function OptionsDropDownValueChanged(app, event)
775     value = app.OptionsDropDown.Value;
776
777     app.Lamp.Color = 'red';
778
779     % sets the row and col labels as well as the file name for
780     % the results based on the default chosen
781     if strcmp(value, 'MakerSpace')
782         app.RowsLabelEditField.Value = 'Students';
783         app.ColsLabelEditField.Value = 'Tools';
784         app.FileEditField.Value = 'MakerSpaceResults.xlsx';
785     elseif strcmp(value, 'Ecosystem')
786         app.RowsLabelEditField.Value = 'Animals';
787         app.ColsLabelEditField.Value = 'Plants';
788         app.FileEditField.Value = 'EcosystemResults.xlsx';
789     elseif strcmp(value, 'Social Network')
790         app.RowsLabelEditField.Value = 'People';
791         app.ColsLabelEditField.Value = 'Events';
792     elseif strcmp(value, 'Transportation Network')
793         app.RowsLabelEditField.Value = 'Locations';
794         app.ColsLabelEditField.Value = 'Routes';
795     elseif strcmp(value, 'Power Grid')
796         app.RowsLabelEditField.Value = 'Suppliers';
797         app.ColsLabelEditField.Value = 'Receivers';
798         app.FileEditField.Value = 'PowerGridResults.xlsx';
799     end
800 end
```

1.3.2.1 Purpose

Generates and Defines the Row and Column labels based on what option is chosen. It can also predefine the results location to save a step later in the export process.

1.3.3 Update Button Pushed

```
802 % Button pushed function: UpdateButton
803 function UpdateButtonPushed(app, event)
804     app.DefaultsPanel.Visible = 'off';
805     app.BiMatPanel.Visible = 'on';
806 end
```

1.3.3.1 Purpose

Hides the defaults panel and reveals the rest of the GUI.

1.3.4 Edit Defaults Button Pushed

```
808 % Button pushed function: EditDefaultsButton
809 function EditDefaultsButtonPushed(app, event)
810     app.BiMatPanel.Visible = 'off';
811     app.DefaultsPanel.Visible = 'on';
812 end
```

1.3.4.1 Purpose

Returns to the Defaults Panel so the user can redefine the Row and Column labels, by hiding the majority of the GUI and only showing the Defaults Panel.

1.3.5 Use a Preformatted Sheet Button Pushed

```
814 % Button pushed function: UseaPreFormatedSheetButton
815 function UseaPreFormatedSheetButtonPushed(app, event)
816 app.MatrixPropertiesPanel.Visible = 'on';
817
818 app.FileNameexTestxlsxEditField.Value = app.FileDropDown.Value;
819
820 % Defines the Ranges based on the premade selections
821 [matrix_range, row_labels_range, col_labels_range] = ConstructPremadeMatrix(app);
822
823 % Shows the Ranges based on the premade selections
824 app.MatrixRangeEditField.Value = matrix_range;
825 app.Row_LabelsRangeEditField.Value = row_labels_range;
826 app.Col_LabelsRangeEditField.Value = col_labels_range;
827 app.TabEditField.Value = app.TabDropDown.Value;
828
829 end
```

1.3.5.1 Purpose

Reveals the Matrix properties Panel and fills in the filename input based on the file dropdown value (B.1.3.12), as well as the tab value based on the tabs dropdown (B.1.3.7). Then defines and displays the ranges using the Construct Predefined Matrix function.

1.3.6 File Dropdown Value Changed

```
831 % Value changed function: FileDropDown
832 function FileDropDownValueChanged(app, event)
833 file = app.FileDropDown.Value;
834 app.FileNameexTestxlsxEditField.Value = file;
835 GenerateTabOptions(app, file);
836
837 % Emptys the Ranges so that user knows to update them with
838 % the "Use Preformated Sheet" button
839 app.MatrixRangeEditField.Value = '';
840 app.Row_LabelsRangeEditField.Value = '';
841 app.Col_LabelsRangeEditField.Value = '';
842
843 app.ShowallPlotsButton.Visible = 'off';
844
845 app.Lamp.Color = 'red';
846
847 end
```

1.3.6.1 Purpose

Generates the tabs options in the tabs dropdown (B.1.3.7) using the Generate Tab Options function (B.1.2.3). It then empties the ranges to prevent operational errors.

1.3.7 Tab Dropdown Value Changed

```
848 % Value changed function: TabDropDown
849 function TabDropDownValueChanged(app, event)
850 value = app.TabDropDown.Value;
851
852 % Emptys the Ranges so that user knows to update them with
853 % the "Use Preformated Sheet" button
854 app.MatrixRangeEditField.Value = '';
855 app.Row_LabelsRangeEditField.Value = '';
856 app.Col_LabelsRangeEditField.Value = '';
857
858 app.ShowallPlotsButton.Visible = 'off';
859
860 app.TabEditField_2.Value = value; % sets the value for the tab to export to
861
862 app.Lamp.Color = 'red';
863
864 end
```

1.3.7.1 Purpose

Sets the tab for the export functions to target and empties the ranges to prevent operational errors.

1.3.8 View Matrix Button Pushed

```
865 % Button pushed function: ViewMatrixButton
866 function ViewMatrixButtonPushed(app, event)
867 [matrix, row_labels, col_labels] = DefineMatrix(app);
868 [~, file] = FilePath(app);
869
870 % defines the tab based on if the tabs checkbox is filled or not
871 if app.TabsCheckBox.Value == 1
872     tab = app.TabEditField.Value;
873 else
874     raw_tabs = sheetnames(file);
875     tab = raw_tabs(1,1);
876 end
877
878 app.MatrixPage = MatrixPage(matrix, row_labels, col_labels, file, tab); % opens and fills the MatrixPage
879
880
881 end
```

1.3.8.1 Purpose

Launches the matrix page so that the user can view the matrix that the program will be using. This includes the matrix and its labels, so it is a great way of checking that the ranges worked correctly.

1.3.8.2 Variables

- Operational Variables
 - matrix - the matrix that represents the Adjacency Matrix defined by the Define Matrix function (B.1.1.2)
 - row_labels - the matrix defined by the Define Matrix function (B.1.1.2)
 - col_labels - the matrix defined by the Define Matrix function (B.1.1.2)
 - file – the filename for the sheet that is being used defined by the File Path function (B.1.1.1)
 - tab – the tab within the excel file that the program is using

1.3.9 List Tabs Button Pushed

```
883 % Button pushed function: ListTabsButton
884 function ListTabsButtonPushed(app, event)
885 [~, file] = FilePath(app);
886 tabs = sheetnames(file);
887 app.ListPage = ListPage(tabs); % Opens the ListPage
888 end
```

1.3.9.1 Purpose

Launches the List page to show the list of tabs defined within the callback. This can be used to verify that the tab selected is in fact within the specified sheet.

1.3.9.2 Variables

- Operational Variables

- file – the filename for the sheet that is being used defined by the File Path function (B.1.1.1)
- tabs – a list of the tabs within the file

1.3.10 Tabs Check Box Value Changed

```

890 % Value changed function: TabsCheckBox
891 function TabsCheckBoxValueChanged(app, event)
892 value = app.TabsCheckBox.Value;
893
894 % Shows the Tab edit field in the "Matrix Properties Panel"
895 % if the the Tab checkbox is filled
896 if value == 1
897 app.TabEditField.Visible = 'on';
898 app.TabEditFieldLabel.Visible = 'on';
899 elseif value == 0
900 app.TabEditField.Visible = 'off';
901 app.TabEditFieldLabel.Visible = 'off';
902 end
903 end

```

1.3.10.1 Purpose

Shows the tabs input in the matrix properties panel if the check box is filled and hides it otherwise.

1.3.11 File Name Value Changing

```

905 % Value changing function: FileNameexTestxlsxEditField
906 function FileNameexTestxlsxEditFieldValueChanging(app, event)
907 %changingValue = event.Value;
908 app.MatrixPropertiesPanel.Visible = 'on';
909 end

```

1.3.11.1 Purpose

Shows the Matrix Properties Panel

1.3.12 File Name Value Changed

```

911 % Value changed function: FileNameexTestxlsxEditField
912 function FileNameexTestxlsxEditFieldValueChanged(app, event)
913 app.MatrixPropertiesPanel.Visible = 'on';
914
915 % Emptys the Ranges so that user knows to update them with
916 % the "Use Preformatted Sheet" button
917 app.MatrixRangeEditField.Value = ' ';
918 app.Row_LabelsRangeEditField.Value = ' ';
919 app.Col_LabelsRangeEditField.Value = ' ';
920
921 app.Lamp.Color = 'red';
922
923 end

```

1.3.12.1 Purpose

Empties the ranges to prevent operational errors.

1.3.13 Run Button Pushed

```
925 % Button pushed function: RunButton
926 function RunButtonPushed(app, event)
927 [path, ~] = FilePath(app);
928
929 % Displays error message if the matrix has not been specified
930 if isempty(app.FileNameexTestxlsxEditField.Value)
931 app.StatusBar.Value = "Specify a Matrix";
932 app.StatusBar.FontWeight = 'bold';
933 app.StatusBar.FontColor = 'red';
934 % Displays error message if the matrix range has not been specified
935 elseif isempty(app.MatrixRangeEditField.Value)
936 app.StatusBar.Value = "Specify a Matrix Range";
937 app.StatusBar.FontWeight = 'bold';
938 app.StatusBar.FontColor = 'red';
939 % Displays error message if the row labels range has not been specified
940 elseif isempty(app.Row_LabelsRangeEditField.Value)
941 app.StatusBar.Value = "Specify a Row Labels Range";
942 app.StatusBar.FontWeight = 'bold';
943 app.StatusBar.FontColor = 'red';
944 % Displays error message if the col labels range has not been specified
945 elseif isempty(app.Col_LabelsRangeEditField.Value)
946 app.StatusBar.Value = "Specify a Col Labels Range";
947 app.StatusBar.FontWeight = 'bold';
948 app.StatusBar.FontColor = 'red';
949 else
950 % Run Analysis
951 app.StatusBar.Value = path;
952 app.StatusBar.FontWeight = 'normal';
953 app.StatusBar.FontColor = 'black';
954 app.Lamp.Color = '0.93,0.69,0.13';
955 drawnow();
956 RunBiMat(app);
957 end
958
959 end
```

1.3.13.1 Purpose

Verifies that the program has inputted a matrix and checks if the ranges are filled out. If not, an error message is displayed in the status bar. If it passes all the checks, then it calls on the Run BiMat function.
(B.1.2.1)

1.3.14 Algorithm Drop Down Value Changed

```
961 % Value changed function: AlgorithmDropDown
962 function AlgorithmDropDownValueChanged(app, event)
963     value = app.AlgorithmDropDown.Value;
964
965     % Shows the KernighanLinTunning Check box if the
966     % LeadingEigenVector Algorithm is chosen
967     if strcmp(value, "LeadingEigenVector")
968         app.KernighanLinTunningCheckBox.Visible = 'on';
969     else
970         app.KernighanLinTunningCheckBox.Visible = 'off';
971     end
972
973     % Reruns the Modularity function for the
974     % algorithm that's chosen
975     Modularity(app);
976
977 end
```

1.3.14.1 Purpose

Reruns the Modularity function (B.1.1.5) with the new algorithm. As well as shows the Kernighan Lin Tunning Check Box if the algorithm is the Leading Eigen Vector.

1.3.15 Show Modules Button Pushed

```
979 % Button pushed function: ShowModulesButton
980 function ShowModulesButtonPushed(app, event)
981     [~, bp_row_labels, bp_col_labels] = DefineBipartiteMatrix(app);
982     [N, ~, ~, ~] = Modularity(app);
983     [matrix, ~, ~] = DefineMatrix(app);
984
985     %% Selecting Algorithm with the Drop Down
986     % LeadingEigenVector
987     if strcmp(app.AlgorithmDropDown.Value, 'LeadingEigenVector') == true
988         algorithm = 'LeadingEigenVector';
989
990     % AdaptiveBrim
991     elseif strcmp(app.AlgorithmDropDown.Value, 'AdaptiveBrim') == true
992         algorithm = 'AdaptiveBrim';
993
994     % LPBrim
995     elseif strcmp(app.AlgorithmDropDown.Value, 'LPBrim') == true
996         algorithm = 'LPBrim';
997     end
998
999     app.ModulesPage = ModulesPage(algorithm, N, matrix, bp_row_labels, bp_col_labels);
1000 end
```

1.3.15.1 Purpose

Launches the Modules Page (B.2) allowing for the user to view the modules and what elements are within them. See section B.2 for the operation of this Page.

1.3.15.2 Variables

- Operational Variables
 - bp_row_labels – the row labels defined by the Define Bipartite Matrix (B.1.1.3)
 - bp_col_labels - the col labels defined by the Define Bipartite Matrix (B.1.1.3)

- algorithm – the algorithm that's used for the modularity functions
- N – the number of modules defined by the Modularity function (B.1.1.5)
- matrix - the matrix that represents the Adjacency Matrix defined by the Define Matrix function (B.1.1.2)

1.3.16 Modules Switch Value Changed

```

1002 % Value changed function: ModulesSwitch
1003 function ModulesSwitchValueChanged(app, event)
1004     app.Lamp.Color = '0.93,0.69,0.13';
1005     drawnow();
1006
1007     % Reruns the FillPandZ function for the new switch value
1008     FillPandZ(app);
1009
1010     app.Lamp.Color = 'green';
1011     drawnow();
1012 end

```

1.3.16.1 Purpose

Reruns the Fill P and Z function (B.1.2.5) based on the value of the modules switch value. (Rows or Columns)

1.3.17 Plot Button Pushed

```

1014 % Button pushed function: PlotButton
1015 function PlotButtonPushed(app, event)
1016     app.Lamp.Color = '0.93,0.69,0.13';
1017     drawnow();
1018
1019     % Displays error message if Anaylisis has not been run yet
1020     if isempty(app.PandZTable.Data)
1021         app.StatusBar.Value = 'Please Run the Analysis';
1022         app.StatusBar.FontWeight = 'bold';
1023         app.StatusBar.FontColor = 'red';
1024     % Opens the PandZPage based on the Modules Switch Value (rows or columns)
1025     else
1026         [Pr, zr, Pc, zc] = PandZ(app);
1027         [%~, row_labels, col_labels] = DefineMatrix(app);
1028         [~, bp_row_labels, bp_col_labels] = DefineBipartiteMatrix(app);
1029
1030
1031         if strcmp(app.ModulesSwitch.Value, 'Rows')
1032             label = app.RowsLabelEditField.Value;
1033             LaunchPandZ(app, bp_row_labels, Pr, zr, label);
1034         elseif strcmp(app.ModulesSwitch.Value, 'Columns')
1035             label = app.ColsLabelEditField.Value;
1036             col_labels = transpose(bp_col_labels); %Converts the column labels into a vertical array
1037             LaunchPandZ(app, col_labels, Pc, zc, label);
1038         end
1039     end
1040
1041     app.Lamp.Color = 'green';
1042     drawnow();
1043 end

```

1.3.17.1 Purpose

Launches the PandZ Page (B.3) with either the Row or column elements. It also verifies that the program has been ran before it tries and opens the program.

1.3.17.2 Variables

- Operational Variables

- Pr – an array of P values for the row elements defined by the PandZ function (B.1.1.8)
- zr – an array of z values for the row elements defined by the PandZ function (B.1.1.8)
- Pc – an array of P values for the column elements defined by the PandZ function (B.1.1.8)
- zc – an array of z values for the column elements defined by the PandZ function (B.1.1.8)
- bp_row_labels – the row labels defined by the Define Bipartite Matrix function (1.1.3)
- bp_col_labels - the column labels defined by the Define Bipartite Matrix function (1.1.3)
- col_labels – the transpose of bp_col_labels so that it can be iterated through and shown properly
- label – the last element of the P and Z pages title defined by the defaults page. It is either the row or column label in the defaults page depending on the modules switch value (1.3.16)

1.3.18 View Nested Matrix Button Pushed

```

1045 % Button pushed function: ViewNestedMatrixButton
1046 function ViewNestedMatrixButtonPushed(app, event)
1047     %[matrix, row_labels, col_labels] = DefineMatrix(app);
1048     [bp, bp_row_labels, bp_col_labels] = DefineBipartiteMatrix(app);
1049     [~, file] = FilePath(app);
1050     [sorted_matrix, ir, ic] = MatrixFunctions.SORT_MATRIX(bp.matrix);
1051
1052     % defines the tab based on if the tabs checkbox is filled or not
1053     if app.TabsCheckBox.Value == 1
1054         tab = app.TabEditField.Value;
1055     else
1056         raw_tabs = sheetnames(file);
1057         tab = raw_tabs(1,1);
1058     end
1059
1060     % sorts the Row and Column labels to align with their original data entries
1061     [M,N] = size(bp.matrix);
1062     sorted_row_labels = strings([M,1]);
1063     sorted_col_labels = strings([N,1]);
1064     for i=1:N
1065         col_index = ic(i);
1066         sorted_col_labels(i,1) = bp_col_labels(1,col_index);
1067     end
1068     for i=1:M
1069         row_index = ir(i);
1070         sorted_row_labels(i,1) = bp_row_labels(row_index,1);
1071     end
1072
1073     app.MatrixPage = MatrixPage(sorted_matrix, sorted_row_labels, sorted_col_labels, file, tab); % opens and fills the MatrixPage
1074
1075 end

```

1.3.18.1 Purpose

Launches the matrix page so that the user can view the nested matrix that the program will be using. This includes the matrix and its labels in the nested order.

1.3.18.2 Variables

- Operational Variables
 - file – the filename for the sheet that is being used defined by the File Path function (B.1.1.1)

- sorted matrix – the sorted Bipartite matrix that is in the most nested form defined by the NestedMatrix function. (B.1.1.7)
- sorted_row_labels – the sorted version of the bp_row_labels. Defined by the NestedMatrix function. (B.1.1.7)
- sorted_col_labels – the sorted version of the bp_col_labels. Defined by the NestedMatrix function. (B.1.1.7)
- tab – the tab within the excel file that the program is using

1.3.19 Add Data Button Pushed

```

1077 % Button pushed function: AddDataButton
1078 function AddDataButtonPushed(app, event)
1079   app.Lamp.Color = '0.93,0.69,0.13';
1080   drawnow();
1081
1082   % Defines the Title of the Combined Results based on if the tabs check box is checked or not
1083   if app.TabsCheckBox.Value == true
1084     Title = app.TabEditField.Value;
1085   else
1086     Title = app.FileNameexTestxlsxEditField.Value;
1087   end
1088
1089   results = ExporttoExcel(app); %Defines the combined results
1090
1091   result = results(3:20, 2); %Defines a temporary array of the most recently ran results
1092   result{1,1} = (Title); % adds the Title name to the temporary result array
1093
1094   app.ComparisonTable.Data = [app.ComparisonTable.Data,result]; % adds temporary result to the data table
1095   app.ComparisonTable.RowName = results(3:20,1); % adds the row labels to the data table
1096
1097   app.Lamp.Color = 'green';
1098   drawnow();
1099 end

```

1.3.19.1 Purpose

Shows the Results for the entire analysis, excluding the P and Z values, into a single column on the compare tab by using the results array created by the Export to Excel function (B.1.2.7). This allows for a quick comparison between different results.

1.3.19.2 Variables

- Operation Variables
 - Title – either the tab or file name of the sheet being used depending on the tab check box value
 - results – a large array that combines all of the results from general properties, Modularity, Nestedness, as well as the P and Z values. Defined by the “Export to Excel” function (B.1.2.7).
 - The results array is shortened to exclude the P and Z values

1.3.20 Generate Button Pushed

```
1107 % Button pushed function: GenerateButton
1108 function GenerateButtonPushed(app, event)
1109 app.Lamp.Color = '0.93,0.69,0.13';
1110 drawnow();
1111
1112 % Generates Plot chosen in the "Plots" dropdown
1113 GeneratePlots(app);
1114
1115 app.Lamp.Color = 'green';
1116 drawnow();
1117 end
```

1.3.20.1 Purpose

Runs the Generate Plots function (B.1.2.2)

1.3.21 Export Button Pushed

```
1119 % Button pushed function: ExportButton
1120 function ExportButtonPushed(app, event)
1121 app.Lamp.Color = '0.93,0.69,0.13';
1122 drawnow();
1123
1124 [bp, ~, ~] = DefineBipartiteMatrix(app);
1125 [path, ~] = FilePath(app);
1126 location = append("BiMat_Analysis\Results\",app.FileEditField.Value);
1127 tab = app.TabEditField_2.Value;
1128
1129 % Displays error message if Analysis has
1130 % not been run yet
1131 if app.NumberOfElementsEditField.Value == 0
1132 app.StatusBar.Value = 'Please Run the Analysis';
1133 app.StatusBar.FontWeight = 'bold';
1134 app.StatusBar.FontColor = 'red';
1135 % Displays error message if the filename for the
1136 % export has not been specified
1137 elseif strcmp(app.FileEditField.Value, '')
1138 app.StatusBar.Value = 'Please Specify a File';
1139 app.StatusBar.FontWeight = 'bold';
1140 app.StatusBar.FontColor = 'red';
1141 % Displays error message if the tab for the
1142 % export has not been specified
1143 elseif strcmp(tab, '')
1144 app.StatusBar.Value = 'Please Specify a Tab';
1145 app.StatusBar.FontWeight = 'bold';
1146 app.StatusBar.FontColor = 'red';
1147 % Exports the Results using the ExporttoExcel function
1148 else
1149 results = ExporttoExcel(app);
1150 writecell(results, location, 'Sheet', tab)
1151 app.StatusBar.Value = path;
1152 app.StatusBar.FontWeight = 'normal';
1153 app.StatusBar.FontColor = 'black';
1154
1155 % Exports all of the Plots to the "Graphs" folder within
1156 % "Results" folder of "BiMat_Analysis"
1157 if app.ExportPlotsCheckBox.Value == true
1158 app.ShowallPlotsButton.Visible = 'on';
```

```

1159 % Modular Graph
1160 plt = "ModularGraph.jpg";
1161 plt_path = append("BiMat_Analysis\Results\Graphs\", plt);
1162 f = figure('Visible','off');
1163 bp.plotter.PlotModularGraph();
1164 exportgraphics(f,plt_path)
1165
1166 % Matrix Plot
1167 plt = "MatrixPlot.jpg";
1168 plt_path = append("BiMat_Analysis\Results\Graphs\", plt);
1169 f = figure('Visible','off');
1170 bp.plotter.PlotMatrix();
1171 exportgraphics(f,plt_path)
1172
1173 % Modular Matrix Plot
1174 plt = "ModularMatrixPlot.jpg";
1175 plt_path = append("BiMat_Analysis\Results\Graphs\", plt);
1176 f = figure('Visible','off');
1177 bp.plotter.PlotModularMatrix();
1178 exportgraphics(f,plt_path)
1179
1180 % Nested Matrix Plot
1181 plt = "NestedMatrixPlot.jpg";
1182 plt_path = append("BiMat_Analysis\Results\Graphs\", plt);
1183 f = figure('Visible','off');
1184 bp.plotter.PlotNestedMatrix();
1185 exportgraphics(f,plt_path)
1186
1187 end
1188
1189 end
1190 app.Lamp.Color = 'green';
1191 drawnow();
1192
1193 end
1194

```

1.3.21.1 Purpose

Writes the results array from the “Export to Excel” function to the excel sheet and tab specified by the file and tab name blanks near the export button. This function also exports the four graphs in this analysis to the “Graphs” folder within the “Results” folder. This is required to view the Plots Page (B.4).

1.3.21.2 Variables

- Operation Variables
 - bp – the bipartite obj defined by the Define Bipartite Matrix Function (1.1.3)
 - path – the combination of the Origin and File defined by the Define File function (B.1.1.1)
 - location – the location where the results will be exported to
 - tab – the tab within the specified sheet that the export will write too
 - results – a large array that combines all of the results from general properties, Modularity, Nestedness, as well as the

P and Z values. Defined by the “Export to Excel” function (B.1.2.7).

- plt – the file name for each individual plot
- plt_path – the location that each individual plot will be saved to
- f – the figure that each plot is plotted into

1.3.22 Show all Plots Button Pushed

```
1196 % Button pushed function: ShowallPlotsButton
1197 function ShowallPlotsButtonPushed(app, event)
1198     [~, file] = FilePath(app);
1199
1200     % defines the tab based on if the tabs checkbox is filled or not
1201     if app.TabsCheckBox.Value == 1
1202         tab = app.TabEditField.Value;
1203     else
1204         raw_tabs = sheetnames(file);
1205         tab = raw_tabs(1,1);
1206     end
1207
1208     app.PlotsPage = PlotsPage(file,tab);
1209
1210 end
```

1.3.22.1 Purpose

Launches the Plots page (B.4) so that the user can view the all of the graphs that the program can create in one window. This Page requires that the export button has been pushed and that they export graphs checkbox has been checked. See section B.4 for how this function is used and operates.

1.3.22.2 Variables

- Operational Variables
 - file – the filename for the sheet that is being used defined by the File Path function (B.1.1.1)
 - tab – the tab within the excel file that the program is using

1.4 Future Work

- Include the Compare Data in the Excel export functions.
- Include the “Modules Page” information in the Excel export functions.

2 Modules Page

Modules

	flower 2	grass 1
insect 2		1
bird 1		1
bird 2	0	1

Show Matrix

Module:

1

[View Module Sizes](#)

3x2

2x2

2.1 Functions

2.1.1 Fill Matrix

```
29 % 
30 - 
31 
32 % Defines the Row and Col Indexes based on which module is selected
33 row_indexes = module_rows{module_num};
34 col_indexes = module_cols{module_num};

35 % Creates Empty Lists for the Row and Col names
36 row_names = string([ ]);
37 col_names = string([ ]);

38 % Selects the Row and Col names based on the Index Values
39 for i=1:length(row_indexes)
40     index = row_indexes(i);
41     label = bp_row_labels(index);
42     row_names(i) = label;
43 end
44 for i=1:length(col_indexes)
45     col_names(1,i) = bp_col_labels(1,col_indexes(i));
46 end
47 
48 app.UITable.Data = int8(module_matrices{module_num}); % Shows Module's Adjacency Matrix
49 app.UITable.ColumnName = col_names; % Shows the sorted Column Names
50 app.UITable.RowName = row_names; % Shows the sorted Row Names
51 
52 end
53 end
54 end
```

2.1.1.1 Purpose

Shows the selected matrix defined by the spinner in the Modules table, and uses the variables defined in the Show Modules Button Pushed callback to do so.

2.1.1.2 Variables

- Input Variables – defined within the Show Matrix Button Pushed Callback (B.2.2.3)
 - module_rows – a cell array of the indexes for each row element within each module
 - module_cols – a cell array of the indexes for each column element within each module
 - module_matrices – a cell array of the adjacency matrices for each module
 - bp_row_labels – the row labels defined by the Define Bipartite Matrix function (B.1.1.3)
 - bp_col_labels - the column labels defined by the Define Bipartite Matrix function (B.1.1.3)
- Operation Variables
 - module_num – the module selected by the module spinner (B.2.2.2)
 - row_indexes – an array of the indexes for each row element within the specified module
 - the specific array is defined by using the module_num to select a specific cell from the module_rows cell array

- col_indexes – an array of the indexes for each column element within the specified module
 - the specific array is defined by using the module_num to select a specific cell from the module_rows cell array
- row_names – an array of the labels for each row element
 - the specific labels are defined by using the row_indexes to define the correct row labels from the bp_row_labels array
- col_names – an array of the indexes for each column element within the specified module from the module_rows cell array
 - the specific labels are defined by using the col_indexes to define the correct column labels from the bp_row_labels array

2.2 Callbacks

2.2.1 Startup Function

```

60 % Code that executes after component creation
61 function startupFcn(app, algorithm, N, raw_matrix, bp_row_labels, bp_col_labels)
62 % Fills the Hidden Edit Fields and Tables that will be used as Global Variables
63 app.AlgorithmEditField.Value = algorithm;
64 app.NEditField.Value = N;
65 app.RowLabels.Data = bp_row_labels;
66 app.ColLabels.Data = bp_col_labels;
67 app.matrix.Data = raw_matrix;
68 end

```

2.2.1.1 Purpose

Defines the Page input variables as global variables by defining them in hidden edit field and tables. If you would like to view the hidden values, go to the component browser, and hide the Modules Table by using the interactivity tab within the inspector. Then show the NEditField, matrix, ColLabels, RowLabels, and AlgorithmEditField using the same method.

2.2.1.2 Variables

- Input Variables
 - algorithm – the selected algorithm within the algorithm drop down in the Modularity Tab defined by Show Modules Button Pushed callback (B.1.3.15)
 - N – the number of modules defined by the Modularity Function (B.1.1.5)
 - bp_row_labels – the row labels defined by the Define Bipartite Matrix function (B.1.1.3)
 - bp_col_labels - the column labels defined by the Define Bipartite Matrix function (B.1.1.3)

- raw_matrix – is the adjacency Matrix defined by the Define Matrix function (B.1.1.2)

2.2.2 Module Spinner Value Changed

```

70 % Value changed function: ModuleSpinner
71 function ModuleSpinnerValueChanged(app, event)
72     % Clears the Data Table
73     app.UITableView.Data = [];
74     app.UITableView.ColumnName = [];
75     app.UITableView.RowName = [];
76 end

```

2.2.2.1 Purpose

Clears the Table whenever the spinner value changes, so the module data can never be mistaken for the wrong module number.

2.2.3 Show Matrix Button Pushed

```

78 % Button pushed function: ShowMatrixButton
79 function ShowMatrixButtonPushed(app, event)
80     % Defines the Global Variables from the Hidden Edit Fields and Tables
81     module_num = app.ModuleSpinner.Value;
82     algorithm = app.AlgorithmEditField.Value;
83     raw_matrix = app.matrix.Data;
84     bp = Bipartite(raw_matrix);
85     N = app.NEditField.Value;
86     bp_row_labels = app.RowLabels.Data;
87     bp_col_labels = app.ColLabels.Data;
88
89     % Selecting Algorithm with the Drop Down
90     % LeadingEigenVector
91     if strcmp(algorithm, 'LeadingEigenVector') == true
92         bp.community = LeadingEigenvector(bp.matrix);
93
94     % ApaditiveBrim
95     elseif strcmp(algorithm, 'AdaptiveBrim') == true
96         bp.community = AdaptiveBrim(bp.matrix);
97
98     % LPBrim
99     elseif strcmp(algorithm, 'LPBrim') == true
100        bp.community = LPBrim(bp.matrix);
101    end
102
103    obj = Detect(bp.community);
104    [module_rows, module_cols] = ExtractCommunityIndexes(obj); % Defines the Indexes of Each Node within each Module
105    module_matrices = ExtractCommunityMatrices(obj); % Defines the Adjancecy Matrix of each Module
106
107    % Verifies that the Selected Moudle Number is actually defined
108    if module_num <= N
109        FillMatrix(app, module_rows, module_cols, module_matrices, bp_row_labels, bp_col_labels);
110    else
111        app.StatusBar.Value = "Selected Module does not exist";
112        app.StatusBar.FontWeight = 'bold';
113        app.StatusBar.FontColor = 'red';
114    end
115 end

```

2.2.3.1 Purpose

Defines the necessary variables for the selected matrix to be shown by the Fill Matrix Function (B.2.1.1). The selected matrix is defined by the spinner in the Modules table. The spinner has a minimum value of 1 and no maximum. However, if the selection is larger than the number of modules an error will be displayed in the status bar.

2.2.3.2 Variables

- Operation Variables
 - module_num – the module selected by the module spinner (B.2.2.2)
 - algorithm – the algorithm that's used for finding modularity defined by the algorithm dropdown defined by the Startup Function (B.2.2.1)
 - N – the number of modules defined by the Startup Function (B.2.2.1)
 - bp_row_labels – the row labels defined by the Startup Function (B.2.2.1)
 - bp_col_labels - the column labels defined by the Startup Function (B.2.2.1)
 - raw_matrix – is the adjacency Matrix defined by the Startup Function (B.2.2.1)
 - bp – the bipartite obj defined by the Define Bipartite Matrix Function (B.1.1.3)
 - bp.community – represents the Bipartite Adjacency Matrix that will be used to calculate modularity
 - will be run in any of the three algorithm scripts: LeadingEigenVector.m, AdaptiveBrim.m, or LPBrim.m depending on the value of the Algorithm Dropdown (B.1.3.14)
 - obj – a variable frame that allows for the Modularity values to be defined based on the correct algorithm
 - module_rows – a cell array of the indexes for each row element within each module
 - module_cols – a cell array of the indexes for each column element within each module
 - module_matrices – a cell array of the adjacency matrices for each module

2.2.4 View Module Size Button Pushed

```
117 % Button pushed function: ViewModuleSizesButton
118 function ViewModuleSizesButton(app, event)
119 % Defines the Global Variables from the Hidden Edit Fields and Tables
120 algorithm = app.AlgorithmEditField.Value;
121 raw_matrix = app.matrix.Data;
122 bp = Bipartite(raw_matrix);
123 N = app.NEditField.Value;
124
125 % Selecting Algorithm with the Drop Down
126 % LeadingEigenVector
127 if strcmp(algorithm, 'LeadingEigenVector') == true
128
129 bp.community = LeadingEigenvector(bp.matrix);
130
131 % ApaptiveBrim
132 elseif strcmp(algorithm, 'AdaptiveBrim') == true
133 bp.community = AdaptiveBrim(bp.matrix);
134
135 % LPBrim
136 elseif strcmp(algorithm, 'LPBrim') == true
137 bp.community = LPBrim(bp.matrix);
138
139
140 obj = Detect(bp.community);
141 module_matrices = ExtractCommunityMatrices(obj); % Defines the Adjancecy Matrix of each Module
142 module_sizes = string([]); % Creates an Empty Array for the sizes to be appended to
143
144 % Converts each cell of the base module_sizes matrix into a string
145 for i=1:N
146 module_sizes(i) = append(string(size(module_matrices{i},1)), 'x', string(size(module_matrices{i},2)));
147 end
148
149 app.ListPage = ListPage(transpose(module_sizes)); % Opens the ListPage
150
151 end
end
```

2.2.4.1 Purpose

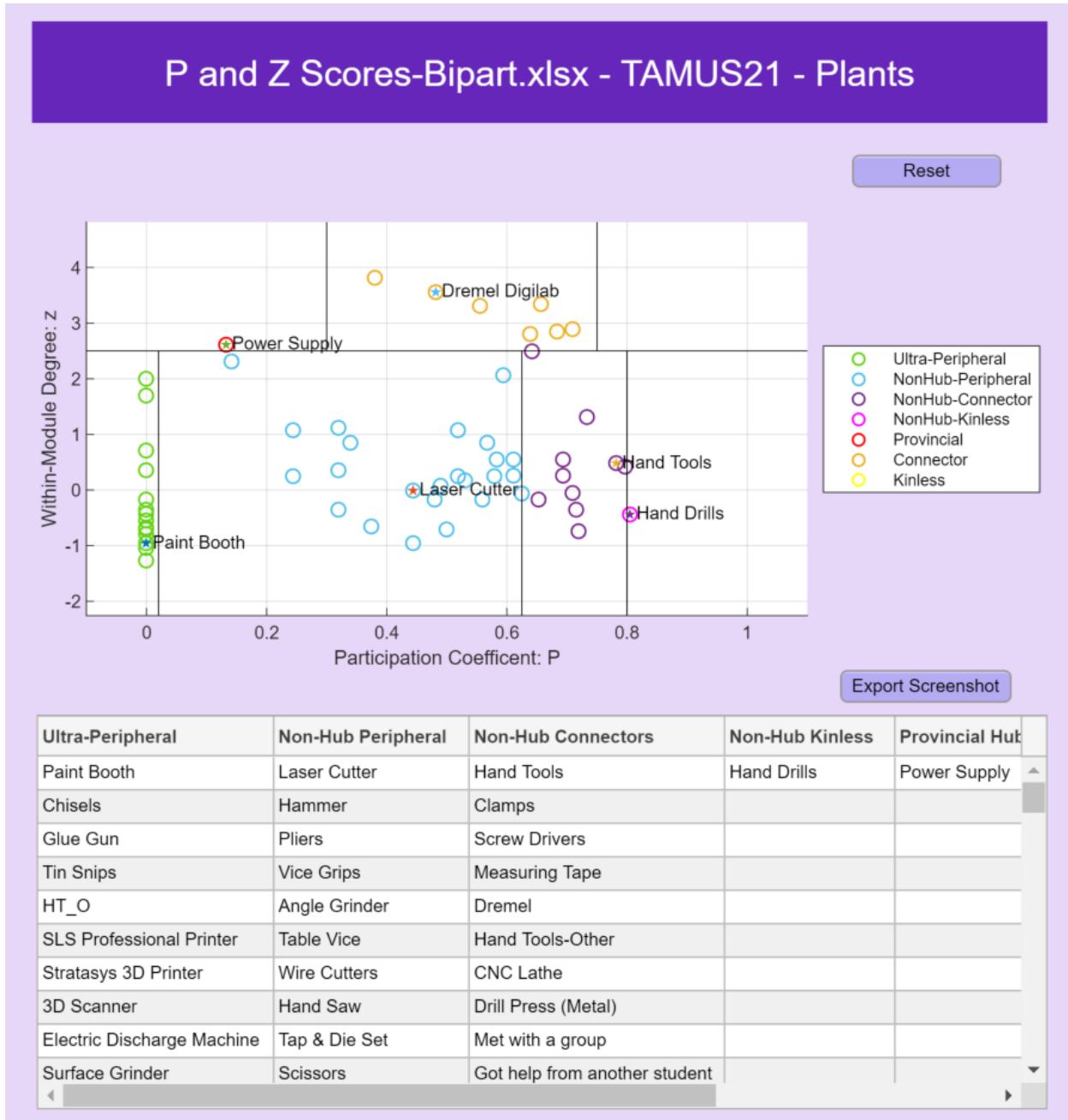
Converts the listed sizes of each module matrix from a cell array of arrays to an array of strings, so that it can be displayed on the List Page (B.1.5)

2.2.4.2 Variables

- Operation Variables
 - algorithm – the algorithm that's used for finding modularity defined by the Startup Function (B.2.2.1)
 - N – the number of modules defined by the Startup Function (B.2.2.1)
 - raw_matrix – is the adjacency Matrix defined by the defined by the Startup Function (B.2.2.1)
 - bp – the bipartite obj defined by the Define Bipartite Matrix Function (B.1.1.3)
 - bp.community – represents the Bipartite Adjacency Matrix that will be used to calculate modularity
 - will be run in any of the three algorithm scripts: LeadingEigenVector.m, AdaptiveBrim.m, or LPBrim.m depending on the value of the Algorithm Dropdown (1.3.14)
 - obj – a variable frame that allows for the Modularity values to be defined based on the correct algorithm

- module_matrices – a cell array of the adjacency matrices for each module
- module_sizes – an array of the raw cells within the module matrices in a string format

3 P and Z Page



3.1 Functions

3.1.1 Plot P and Z

```
24 function PlotPandZ(app, labels, p, z, roles)
25
26 N = length(p); %Sets the length of the iteration
27
28 % Intializes the empty grouped data arrays
29 ultra_peripheral = strings([N,4]);
30 peripheral = strings([N,4]);
31 nonhub_connector = strings([N,4]);
32 nonhub_kinless = strings([N,4]);
33 provincial = strings([N,4]);
34 connector = strings([N,4]);
35 kinless = strings([N,4]);
36
37 % Puts the Data into Groups based on its role
38 for i=1:N
39     % Non-Hub Roles
40     if z(i) < 2.5
41         if p(i) <= .1
42             ultra_peripheral(i,1) = labels(i);
43             ultra_peripheral(i,2) = p(i);
44             ultra_peripheral(i,3) = z(i);
45             ultra_peripheral(i,4) = "NonHub-Ultra-Peripheral";
46         elseif p(i) >= .1 && p(i) <= .625
47             peripheral(i,1) = labels(i);
48             peripheral(i,2) = p(i);
49             peripheral(i,3) = z(i);
50             peripheral(i,4) = "NonHub-Peripheral";
51         elseif p(i) >= .625 && p(i) <= .8
52             nonhub_connector(i,1) = labels(i);
53             nonhub_connector(i,2) = p(i);
54             nonhub_connector(i,3) = z(i);
55             nonhub_connector(i,4) = "NonHub-Connector";
56         elseif p(i) >= .8
57             nonhub_kinless(i,1) = labels(i);
58             nonhub_kinless(i,2) = p(i);
59             nonhub_kinless(i,3) = z(i);
60             nonhub_kinless(i,4) = "NonHub-Kinless";
61     end
```

```

62 % Hub Roles
63 - elseif z(i) >= 2.5
64 -   if p(i) <= .3
65 -     provincial(i,1) = labels(i);
66 -     provincial(i,2) = p(i);
67 -     provincial(i,3) = z(i);
68 -     provincial(i,4) = "Hub-Provincial";
69 -   elseif p(i) >= .3 && p(i) <= .75
70 -     connector(i,1) = labels(i);
71 -     connector(i,2) = p(i);
72 -     connector(i,3) = z(i);
73 -     connector(i,4) = "Hub-Connector";
74 -   elseif p(i) >= .75
75 -     kinless(i,1) = labels(i);
76 -     kinless(i,2) = p(i);
77 -     kinless(i,3) = z(i);
78 -     kinless(i,4) = "Hub-Kinless";
79 -   end
80 - end
81 -
82 sz = 50; %Marker Size
83 -
84 % Plotting Each Subgroup only when there is data within the generated frame
85 if isempty(ultra_peripheral) == 0
86   x = str2double(ultra_peripheral(:,2)); % converts the strings from the subgroup arrays into usable numbers
87   y = str2double(ultra_peripheral(:,3)); % converts the strings from the subgroup arrays into usable numbers
88   plot = scatter(app.UIAxes, x, y, sz,[0.39,0.83,0.07], 'LineWidth', 1);
89   plot.DataTipTemplate.DataTipRows(1) = dataTipTextRow('Node:',ultra_peripheral(:,1));
90   plot.DataTipTemplate.DataTipRows(2) = dataTipTextRow('P:',ultra_peripheral(:,2));
91   plot.DataTipTemplate.DataTipRows(3) = dataTipTextRow('z:',ultra_peripheral(:,3));
92   plot.DataTipTemplate.DataTipRows(4) = dataTipTextRow('Role:',ultra_peripheral(:,4));
93   hold(app.UIAxes, 'on'); % Allows for the next group to be plotted on the same axis
94
95 end
96
97
162 lower_bound = min(z) - 1; % Defines the lower y-limit based on the minimum z value
163 upper_bound = max(z) + 1; % Defines the upper y-limit based on the maximum z value
164
165 xlim(app.UIAxes, [-0.1,1.1]) % Applies the x-limits to the plot
166 ylim(app.UIAxes, [lower_bound, upper_bound]) % Applies the y-limits to the plot
167
168 % Plotting the asymptotes that separate the subgroups
169 line(app.UIAxes, [-0.1,1.1],[2.5,2.5], 'color', 'black');
170 line(app.UIAxes, [0.02,0.02],[lower_bound,2.5], 'color', 'black');
171 line(app.UIAxes, [0.3,0.3],[2.5,upper_bound], 'color', 'black');
172 line(app.UIAxes, [0.625,0.625],[lower_bound,2.5], 'color', 'black');
173 line(app.UIAxes, [0.8,0.8],[lower_bound,2.5], 'color', 'black');
174 line(app.UIAxes, [0.75,0.75],[2.5,upper_bound], 'color', 'black');
175
176 % Defines the Legend's information and location
177 legend(app.UIAxes,["Ultra-Peripheral", "NonHub-Peripheral", "NonHub-Connector", "NonHub-Kinless", "Provincial", "Connector", "Kinless"], 'on'); % allows for the cell Sections to be added to the same axes
178
179
180
181 app.roleTable.Data = roles; % fills in the role table for Cells to be selected
182
183 end

```

3.1.1.1 Purpose

Both fills the Roles table and plots the grouped P and Z data. The table is filled with the “roles” input variable. The Plotting functions are split up by the element defined roles so that the colors could be different. It also defines the “DataTips” for each subgroup, which is the label that shows when hovering over the data point. As well as plots the lines that operate the roles based on the values that define the roles.

3.1.1.2 Variables

- Input Variables
 - P – an array of P values
 - z – an array of z values
 - labels – an array of element names
 - roles – an array of the elements, where each column represents the elements that share a role
- Operation Variables
 - ultra_peripheral – an array of all the elements that are ultra-peripherals. The columns represent- (1. Label, 2. P value, 3. Z value, 4. Role)

- peripheral – an array of all the elements that are peripherals. The columns represent- (1. Label, 2. P value, 3. Z value, 4. Role)
- nonhub_connector – an array of all the elements that are nonhub_connectors. The columns represent- (1. Label, 2. P value, 3. Z value, 4. Role)
- nonhub_kinless – an array of all the elements that are nonhub_kinless. The columns represent- (1. Label, 2. P value, 3. Z value, 4. Role)
- provincial – an array of all the elements that are provincials. The columns represent- (1. Label, 2. P value, 3. Z value, 4. Role)
- connector – an array of all the elements that are connectors. The columns represent- (1. Label, 2. P value, 3. Z value, 4. Role)
- kinless – an array of all the elements that are kinless. The columns represent- (1. Label, 2. P value, 3. Z value, 4. Role)
- role_col –
- N – the length of the P values array, which sets the length of iteration
- x – an array of the 2nd column of the role arrays, that has been converted to numbers rather than strings
- y – an array of the 3rd column of the role arrays, that has been converted to numbers rather than strings

3.1.2 Reset Plot Labels

```

186 %>>> 
187 % These Values are Defined behind the roles table and they are made invisible in the startupfcn()
188 -
189 -
190 -
191 -
192 -
193 -
194 -
195 -
196 -
197 -
198
function ResetPlotLabels(app)
    % These Values are Defined behind the roles table and they are made invisible in the startupfcn()
    labels = app.labelsTable.Data;
    p = app.PTable.Data;
    z = app.zTable.Data;
    roles = app.rolesTable.Data;
    role_list = app.role_col.Data;

    cla(app.UIAxes) % Clears figure

    PlotPandZ(app, labels, p, z, roles); % Replots data
end

```

3.1.2.1 Purpose

Clears the figure and the replots the data, so all the cell selection labels disappear.

3.1.2.2 Variables

- Operation Variables
 - p – an array of P values defined by the startup function (B.3.2.1)
 - z – an array of z defined by the startup function (B.3.2.1)

- labels – an array of element names defined by the startup function (B.3.2.1)
- roles – an array of the elements placed into the column that represents the different roles defined by the startup function (B.3.2.1)
- role_list – a column array that lists the elements role in the order that the labels are originally defined by the startup function (B.3.2.1)

3.2 Callbacks

3.2.1 Startup Function

```

204 % Code that executes after component creation
205 function startupFcn(app, file, tab, labels, p, z, label, roles, role_col)
206 app.Label.Text = append('P and Z Scores', '-', file, ' - ', tab, ' - ', label);
207
208 app.labelsTable.Visible = 'off';
209 app.labelsTable.Data = labels;
210 app.PTable.Visible = 'off';
211 app.PTable.Data = p;
212 app.zTable.Visible = 'off';
213 app.zTable.Data = z;
214 app.role_col.Visible = 'off';
215 app.role_col.Data = role_col;
216
217 PlotPandZ(app, labels, p, z, roles);
218 end

```

3.2.1.1 Purpose

Defines the Page input variables as global variables by defining them in hidden edit field and tables. If you would like to view the hidden values, go to the component browser, and hide the roles Table by using the interactivity tab within the inspector. Then show the role_col, labelsTable, zTable, and PTable using the same method.

3.2.1.2 Variables

- Function Inputs
 - file – the filename for the sheet that is being used
 - tab - the tab within the specified sheet that the function will be using
 - p – an array of P values either for the row or column elements based on the modules switch (B.1.3.16)
 - z – an array of z values either for the row or column elements based on the modules switch (B.1.3.16)
 - labels – an array of element names from either the row or column elements depending on the modules switch value (B.1.3.16)
 - label – the last element of the P and Z pages title defined by the defaults page. It is either the row or column label in the defaults page depending on the modules switch value (B.1.3.16)

- roles – an array of the elements placed into the column that represents the different roles
- role_col – a column array that lists the elements role in the order that the labels are originally in

3.2.2 Roles Table Cell Selection

```

220 % Cell selection callback: rolesTable
221 function rolesTableCellSelection(app, event)
222   cell = event.Indices;
223
224   labels = app.labelsTable.Data;
225   p = app.PTable.Data;
226   z = app.zTable.Data;
227   role_list = app.role_col.Data;
228   role_matrix = app.rolesTable.Data;
229
230   selected_cell = role_matrix(cell(1,1),cell(1,2));
231
232   N = length(labels);
233   label = strings([N,1]);
234   role = strings([N,1]);
235
236   % Check for the selected cell within the original data and re-plots it with different parameters
237   for i=1:N
238     if strcmp(selected_cell,labels(i))
239
240       label(:,1) = labels(i);
241       role(:,1) = role_list(i);
242
243       pt = scatter(app.UIAxes, p(i), z(i), 'filled','p'); % Plots
244       pt.DataTipTemplate.DataTipRows(1) = dataTipTextRow('Node:',label); %Hovering Label row 1
245       pt.DataTipTemplate.DataTipRows(2) = dataTipTextRow('P:',p(i)); %Hovering Label row 2
246       pt.DataTipTemplate.DataTipRows(3) = dataTipTextRow('z:',z(i)); %Hovering Label row 3
247       pt.DataTipTemplate.DataTipRows(4) = dataTipTextRow('Role:',role); %Hovering Label row 4
248       hold(app.UIAxes, 'on');
249
250       text(app.UIAxes, p(i)+0.01,z(i)+0.02,labels(i)); %Fixed Label
251       hold(app.UIAxes, 'on');
252
253     end
254   end
255 end

```

3.2.2.1 Purpose

When the user clicks on a cell within the roles table, the data point is highlighted by both the fixed and hovering data labels. These values can be removed by hitting the rest button (B.3.2.3).

3.2.2.2 Variables

- Input Variables
 - event – the selected cell's value
- Operation Variables
 - p – an array of P values from either the row or column elements depending on the modules switch value (B.1.3.16)
 - z – an array of z values from either the row or column elements depending on the modules switch value (B.1.3.16)

- labels – an array of element names from either the row or column elements depending on the modules switch value (B.1.3.16)
- role_list – a column array that lists the elements role in the order that the labels are originally in from either the row or column elements depending on the modules switch value (B.1.3.16)
- role_matrix – an array of the elements placed into the column that represents the different roles. They are either the row or column elements depending on the modules switch value (B.1.3.16)
- label – a reordered version of the labels array, so that the cell selection shows the correct role value (B.1.3.16)
- role – a reordered version of the role_list array, so that the cell selection shows the correct role
- N – the length of labels, which sets the length of the iterative calculation

3.2.3 Reset Button Pushed

```

257 % Button pushed function: ResetButton
258 function ResetButtonPushed(app, event)
259     ResetPlotLabels(app);
260 end

```

3.2.3.1 Purpose

The Reset button clears all of the labels created by the cell selection callback (B.3.2.2). It does so by calling on the Reset Plot Labels Function (B.3.1.2).

3.2.4 Export Screenshot Button Pushed

```

262 % Button pushed function: ExportScreenshotButton
263 function ExportScreenshotButtonPushed(app, event)
264     filename = app.Label.Text;
265     path = "BiMat_Analysis\Results\P and Z\";
266     location = append(path, filename, ".jpg");
267     exportapp(app.UIFigure,location)
268 end
269 end

```

3.2.4.1 Purpose

Saves a screenshot of the app in its current status within the “P and Z” folder within the “Results” folder of BiMat_Analysis.

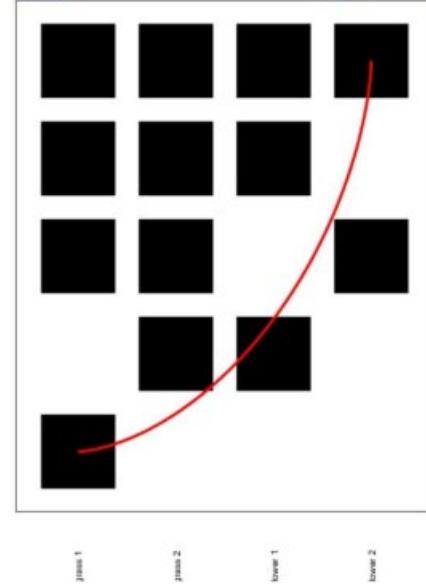
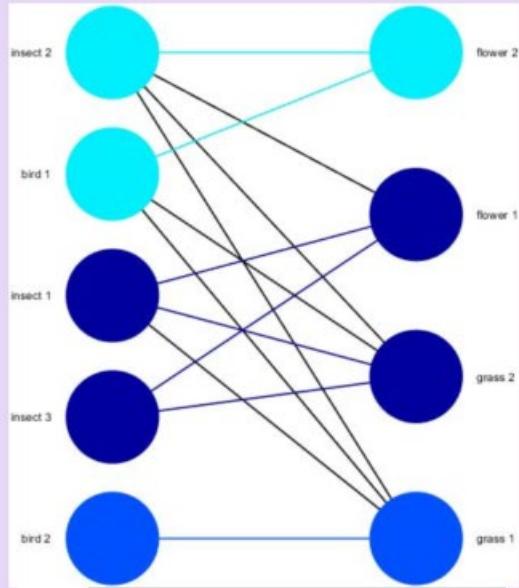
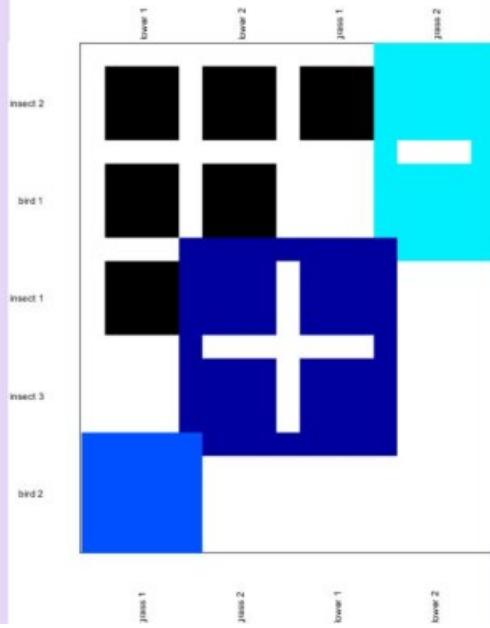
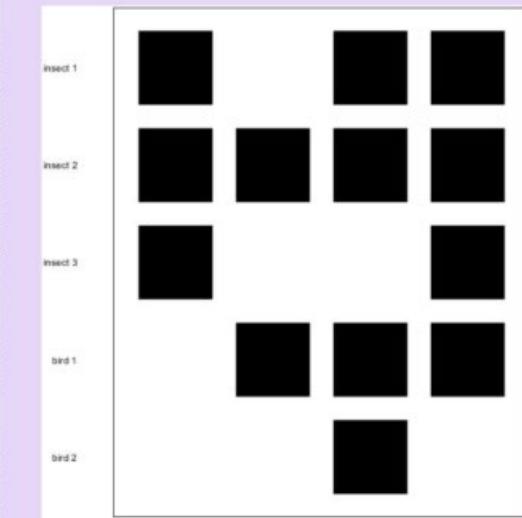
- This screenshot can also capture the labels shown by the cell selection callback (B.3.2.2).
- The filename will be the title that the P and Z page shows at the top of its window.

3.2.4.2 Variables

- Operation Variables
 - filename – the filename that the screenshot will be saved as
 - path – the path that the file will be save with
 - location – the appended values of the filename, path, and file extension

4 Plots Page

Test.xlsx - Test1



```

1 classdef PlotsPage < matlab.apps.AppBase
2
3 % Properties that correspond to app components
4 properties (Access = public)
5 - UIFigure matlab.ui.Figure
6 - ModularMatrix matlab.ui.control.Image
7 - NestedMatrix matlab.ui.control.Image
8 - Matrix matlab.ui.control.Image
9 - ModularGraph matlab.ui.control.Image
10 - Label matlab.ui.control.Label
11 end
12
13 % Callbacks that handle component events
14 methods (Access = private)
15
16 % Code that executes after component creation
17 function startupFcn(app, file, tab)
18 app.Label.Text = append(file, ' - ', tab); % Shows Title
19 end
20
21 % Component initialization
22 methods (Access = private) ...
23
24 % App creation and deletion
25 methods (Access = public) ...
96 end

```

4.1 Callbacks

4.1.1 Startup Function

4.1.1.1 Purpose

This function shows the title of the matrix page by appending the file name and tab name. Then it displays the four graphs located in the “Graphs” within the “Results” folder. These four graphs are saved in this location with the Export button (B.1.3.21) when the export plots checkbox is checked.

4.1.1.2 Variables

- Input Variables
 - file – the filename for the sheet that is being used
 - tab - the tab within the specified sheet that the program is analyzing

5 Matrix Page

Test.xlsx - Test1

	flower 1	flower 2	grass 1	grass 2
insect 1		1	0	1
insect 2		1	1	1
insect 3		1	0	0
bird 1		0	1	1
bird 2		0	0	1

```
1 classdef MatrixPage < matlab.apps.AppBase
2
3 % Properties that correspond to app components
4 properties (Access = public)
5 - UIFigure matlab.ui.Figure
6 - MatrixTable matlab.ui.control.Table
7 - Label matlab.ui.control.Label
8 end
9
10 methods (Access = private)
11
12 function FillFigures(app, matrix, row_labels, col_labels)
13
14 app.MatrixTable.Data = matrix; % Shows Adjacency Matrix
15 app.MatrixTable.ColumnName = col_labels; % Shows Column Names
16 app.MatrixTable.RowName = row_labels; % Shows Row Names
17 end
18 end
19
20 % Callbacks that handle component events
21 methods (Access = private)
22
23 % Code that executes after component creation
24 function startupFcn(app, matrix, row_labels, col_labels, file, tab)
25
26 FillFigures(app, matrix, row_labels, col_labels); % Calls the functions that shows the data
27
28 app.Label.Text = append(file, ' - ',tab); % Shows Matrix Title
29 end
30 end
31
32 % Component initialization
33 methods (Access = private) ...
34
35 % App creation and deletion
36 methods (Access = public) ...
37 end
```

5.1 Functions

5.1.1 Fill Figures

5.1.1.1 Purpose

This function shows the matrix, row_labels, and col_labels by displaying them in the MatrixTable. The matrix can be any matrix that's defined by the startup function (B.4.2.1). In this program it is used to show both the adjacency, modules, and nestedness matrix.

5.1.1.2 Variables

- **Input Variables**
 - matrix – the matrix data that is being shown in the Matrix Page
 - row_labels – the row element labels for the matrix that is being shown
 - col_labels - the column element labels for the matrix that is being shown

5.2 Callbacks

5.2.1 Startup Function

5.2.1.1 Purpose

This function shows the title of the matrix page by appending the file name and tab name. Then calls on the Fill Matrix function to show the data (B.4.1.1). This matrix and its label can be anything as long as they are defined when opening this app from another script.

5.2.1.2 Variables

- **Input Variables**
 - file – the filename for the sheet that is being used
 - tab - the tab within the specified sheet that the program is analyzing
 - matrix – the matrix data that is being shown in the Matrix Page
 - row_labels – the row element labels for the matrix that is being shown
 - col_labels - the column element labels for the matrix that is being shown

6 List Page

```

1 classdef ListPage < matlab.apps.AppBase
2
3     % Properties that correspond to app components
4     properties (Access = public)
5         UIFigure    matlab.ui.Figure
6         ListTable   matlab.ui.control.Table
7     end
8
9     % Callbacks that handle component events
10    methods (Access = private)
11
12        % Code that executes after component creation
13        function startupFcn(app, list)
14            app.ListTable.Data = list; %Shows the List defined as a input variable on the page
15        end
16    end
17
18    % Component initialization
19    methods (Access = private) ...
20
21    % App creation and deletion
22    methods (Access = public) ...
23
24 end

```

6.1 Callbacks

6.1.1 Startup Function

6.1.1.1 Purpose

Shows the list defined as an input to the function within the ListTable.

6.1.1.2 Variables

- Input Variables
 - list – a column array of values

7 *Name*MatrixProperties.m

7.1 Functions

```

function [matrix_range, row_labels_range, col_labels_range, tabs] = BipartMatrixProperties(tab_name, file)
    raw_tabs = sheetnames(file);
    tabs = raw_tabs;

    if strcmp(tab_name, 'GTF19') == true
        matrix_range = 'B2:W160';
        row_labels_range = 'A2:A160';
        col_labels_range = 'B163:W163';
    elseif strcmp(tab_name, 'GTF20') == true
        matrix_range = 'B3:Q66';
        row_labels_range = 'A3:A66';
        col_labels_range = 'B2:Q2';
    elseif strcmp(tab_name, 'TAMUF20') == true
        matrix_range = 'B3:W52';
        row_labels_range = 'A3:A52';
        col_labels_range = 'B2:W2';
    end

```

7.1.1 *Name*MatrixProperties

7.1.1.1 Purpose

Defines the ranges for individual tabs within a specific sheet, so that the user does not have to manually fill out the ranges inputs within the Matrix Properties Panel.

- Each Preformatted sheet needs to have its own *Name*MatrixProperties script

7.1.1.2 Variables

- Input Variables
 - tab_name – the tab name that the program is using
 - since this script is for premade matrices, the tab name will be the value shown in the tabs drop down. (B.1.3.7)
 - file – the filename for the sheet that is being used
 - since this script is for premade matrices, the file name will be the value shown in the file drop down. (B.1.3.6)
- Output Variables
 - matrix_range - the cells that contain the actual data for the Adjacency Matrix within a specific tab
 - row_labels_range - the cells that contain the row labels for the Adjacency Matrix within a specific tab
 - col_labels_range - the cells that contain the column labels for the Adjacency Matrix within a specific tab