Getting Started with Mathematica

Mathematica has a unique interface that takes a while to get used to. You open to a blank page, like a word processor, where you can type anything you want. Most of the time you will type commands that do things with your data: connect to databases, plot graphs, carry out calculations. Unlike other statistical or mathematical programs, the commands you type and the output you get remain on the page, which gives you a record of what you’ve done step-by-step. To help organize your work, you can add headers, format boxes, etc. with the Format Menu.

Cells

Cells are an important organizing feature of Mathematica. Note the “cell” markers on the right margin. Each cell is bounded by a bracket and commands within a cell are executed together. You can open and close cells by double clicking the bracket. This can be useful if you have lots of stuff in a notebook… you can give a section a heading, which causes cells in that section to be grouped, after which you can close the section by double clicking.

Shift + Enter causes a cell to be executed. Pressing the enter key creates a new line, just like in a word processor, but when you want to execute a command you typed, you type SHIFT+ENTER somewhere in the cell and all commands in the cell are executed.

Commands (aka Functions)

Mathematica is designed to be as easy to learn as possible so that you can concentrate on working instead of the program. Almost all commands are English words written out in full with capitals at the beginning of words and brackets [] at the end of the command. For example, the command to calculate an average of a set of numbers is Mean[], the command to do a principal components analysis is PrincipalComponents[], and the command to take the logarithm of a number is Log[].

See the “Mathematica Quick Guide” handout for some especially useful functions.

Formatting Output

Mathematica is clever about how it provides output and it tries to keep the results as accurate as possible. For example, if you calculate the average of the following numbers

Mean[ {1, 5, 10, 3, 20, 40} ]
the answer extends to many decimal places, so Mathematica reports it more precisely as a fraction: 79/6. You may want an ordinary number, however, and you can force Mathematica to format its output the way you want:

\[
\text{Mean\{1, 5, 10, 3, 20, 40\} \.OutputForm}
\]

This command now gives you 13.1667. Another useful formatting function is \[\text{MatrixForm}\\], which causes a table to be displayed neatly in columns instead of wrapping around the page.

**Packages**

Many of Mathematica's functions are available at startup, but additional specialized functions are available from add-in packages. We will use a number of functions from built in packages, plus some from a customized package written specially for this course which you will have to install.

You can load a built-in or installed package in two ways, with the Needs[] function or with the symbols “<<”. The package name has quotation marks if you use the Needs[] function, but does not with “<<”. Package names are always indicated with a backwards apostrophe at the end of the name.

\[\text{Needs\{"PackageName`"\}}\]

\[<<\text{PackageName}`\]

**Graphics**

Mathematica is good at graphics. You can either use simple functions like \[\text{ListPlot}\] to create a generic graph, or you can experiment with \[\text{Graphics}\] to create a completely customized graphic.

**Getting Help**

The Documentation Center under the Help Menu is invaluable. You can read the documentation like a book, or you can search for information you need. Because Mathematica functions are named using complete English words with the first letter capitalized, you can often guess what a function name might be: “Plot”, “Mean”, “Median”, “LinearRegression”. If your guess is wrong, the documentation usually lets you find the right function quickly. For example, “LinearRegression” isn’t a Mathematica function, but you are quickly led to “LinearModelFit”.

- Note the “See Also” and “Tutorial” sections at the bottom of most help pages.
- The Function Navigator is also a useful tool that you can use to find functions by browsing categories.
Programming in Mathematica

You can do all sorts of things without programming, but writing code can help you automate tasks. You can accomplish a lot with two very simple programming components: loops and conditional statements.

Loops

Loops are programming structures that repeat something many times, such as counting the number of taxa in many different time bins. The basic syntax of a loop is that it contains a starting statement, followed by code that gets repeated, then an iterator that specifies how many times to perform the loop and, often, creates a special counting variable that can be used to change what happens in the code from one repetition to the next. The Table[] function is the most common kind of loop in Mathematica. The function is called “Table” because the results of each repetition are returned in a separate cell of a table. Please look at the Mathematica documentation for Table[].

Conditional Statements

It is often useful, especially in a loop, to check whether something is true before doing it. For example, you might want to drop small samples from your data: you could write a loop that looks at the size of each sample and drops the ones that have fewer than five items. Mathematica has several conditional functions, the most common of which is If[]. The If function takes three arguments: the first is the statement that needs checking, the second is the code to execute if the statement is true, the third is the code to execute if the statement is false. Please look at the Mathematica documentation for If[].

Querying MySQL from Mathematica

You can load data directly into Mathematica from MySQL using Database Link functions. This gives you the power to select only the data you want.

Instructions for First Time Connection

<<DatabaseLink`

Next, set up a database link and give it a name (“conn” in this example):

conn = OpenSQLConnection[]

This function will open a Connection Tool window, where you can create a new connection to your mySQL database. Click “New” and follow the steps to create the connection:
1. give it a short name, e.g. ‘mySQL’  
2. specify System Level;  
3. choose “MySQL(Connector/J)” from the database type menu;  
4. Specify “localhost” or “129.0.0.1”, port 3306, username “root”, password for your mySQL (probably “root” by default).  
   [test it to make sure it works];  
5. store password if you wish

Close the connection again...

CloseSQLConnection[conn]

**Normal Connections**

Once you have created a connection, you can use it again without going through the configuration steps:

```mathematica
In[22]:= conn = OpenSQLConnection["MySQL", Catalog -> "Felidae", Username -> "root", Password -> "root"]
CloseSQLConnection[conn]

Out[22]= SQLConnection[mySQL, 2, Open, Catalog -> Felidae]
Out[23]= {} 
```

This opens the database (aka, “Catalog”) named “Felidae” using the stored connection named “mySQL”.

Put semicolons at the end of the line to suppress the output:

```mathematica
In[24]:= conn = OpenSQLConnection["MySQL", Catalog -> "Felidae", Username -> "root", Password -> "root"];
CloseSQLConnection[conn];
```

Always close the connection when you are finished with it.

But there’s no use opening a database connection, unless you do something with it. We’ll use the connections to load data into Mathematica using an SQL query. The SQLExecute[] function allows you to send an SQL statement to the database and get back the results. Store the results in a variable so you can use them:

```mathematica
In[32]= conn = OpenSQLConnection["MySQL", Catalog -> "Felidae", Username -> "root", Password -> "root"]; 
data = SQLExecute[conn, "Select * FROM Occurrences;"];
CloseSQLConnection[conn];
```

The entire occurrences table should now be stored in the variable data. We can check the first line to make sure:
In[35]= data[[1]]

Out[35]= {1, 11797, Carnivora, Felidae, Vishnuferis, sp., H. O'Regan, Pilgrim, 1932, 4195, Ramanagar, India, Jammu and Kashmir, Ramanagar, 32.8167, 75.36667000000010, Neogene, Miocene, Null, Null, Null, 10., 10., 9.999999999999999, Chinji, Null}