Mathematica
An Introduction
Uses for **Mathematica**

Calculations, simple or complicated

\[
\begin{align*}
\text{In[1]} &= 12 + 20 \\
\text{Out[1]} &= 32 \\
\text{In[2]} &= \frac{((10 \cdot 5) + (20 \cdot 30))}{10^{0.5}} \\
\text{Out[2]} &= 205.548
\end{align*}
\]

Mathematical functions

\[
\text{In[7]} = \log_{25} \text{ Out[7]} = 3.21888
\]

Statistical analysis

\[
\begin{align*}
\text{Out[15]} &= \{\text{ANOVATable} \rightarrow \{
\text{DF}, \text{SumOfSq}, \text{MeanSq}, \text{FRatio}, \text{PValue}
\}
\}
\end{align*}
\]

Programming, simple or complicated

\[
\text{In[3]} = \text{Do[Print["Species " <> ToString[x], \{x, 8\}]
\}
\]

\[
\begin{align*}
\text{Species 1} \\
\text{Species 2} \\
\text{Species 3} \\
\text{Species 4} \\
\text{Species 5} \\
\text{Species 6} \\
\text{Species 7} \\
\text{Species 8}
\end{align*}
\]

\[
\text{PhyloTmp} = \text{Table[}
\begin{align*}
\text{Flatten[\{\text{fau}\{x\}, \{1 \times \{1, 3, 4\}\},}
\text{MeanPairwiseDivergence[ToString[\#] \& /@ fau\{x\}, \{1 \times \{2\}, \text{Varg}\}],}
\text{\{\text{x}, \text{Length[fau\{x\}]\}};}
\text{PhyloData = Select[PhyloTmp, \#[[4]] > 2 \&\& \#[[5]] > 12 \&\& \#[[5]] < 39 \&\&}
\text{Do[PhyloData[\{\text{x}, \{1 \times \{2\}\},}
\text{GeoGridPosition[GeoPosition[\{PhyloData[\{\text{x}, \{3\}\}, PhyloData[\{\text{x}, \{2\}\}],}
\text{\{"LambertAzimuthal","Centering" -> \{70, -100\}\}\}\}}],
\text{\{\text{x}, \text{Length[PhyloData]\}}
\text{Graphics[}
\begin{align*}
\text{\{(LightGray, PointSize[0.025], Point[Flatten[\{randommeananddivergence, 1\}],
\text{\{Red, PointSize[0.015], Point[PhyloData[\{1 \times \{4, 5\}\}], Frame -> True,}
\text{ AspectRatio -> 1 / GoldenRatio]\}}
\end{align*}
\]
Graphics in *Mathematica*

**Plots of data**

\[\text{Out[1]} = \text{ListPlot[data]}\]

**Three-dimensional plots**

\[\text{Out[2]} = \text{Plot3D[Sin[x] + Cos[y], \{x, 0, 100\}, \{y, 0, 100\}]}\]

**Plots of functions**

\[\text{Out[3]} = \text{Plot[Log[x], \{x, 0, 1000\}]}\]

**Specialized objects**

\[\text{Out[4]} = \text{PolyhedronData["Dodecahedron"]}\]
Getting help

Mathematica help files can be browsed or searched from the Documentation Center of the Help menu.

Function names are always made of up complete words, no spaces, with the first word capitalized.

Search for functions you hope exist: “Histogram”, “LinearRegression”, “PrincipalComponents”, “GenomeData”

Note Function Browser and Mathematica Book help buttons at top left of the Documentation Center.

Kernels and Notebooks

Mathematica has two components, the \textit{kernel} and the \textit{notebook}

The \textit{kernel} is the invisible part of the program that does all the calculations.

The \textit{notebook} is the main user interface, its purpose is to allow you to perform analyses and to save them for re-use or for later reference.

You can work with many notebooks at once. They share information between them because they interface with the same kernel.

For advanced work you can work with two kernels, which allows you to run two sets of calculations in different notebooks at the same time.
Notebooks and cells

Notebooks are organized into **cells**

Default cells are for calculations, with input entered by you followed by output created by the kernel.

Cells must be executed to obtain output: **Shift + Enter to execute**

Cells may be executed more than once, and the input can be changed between executions.

**Brackets** in the right margin show cell boundaries and distinguish between input and output.

**Uses for brackets:**

1. monitor calculations (bracket is highlighted while the kernel is executing)
2. select entire cell for deletion
3. hide output by double clicking
Formatting notebooks

Notebooks can be formatted like a word processor document.

Individual cells can be formatted as titles, text, section headings, or input (input is the default).

Use Format | Style menu to format individual cells.

Use Format | Stylesheet menu to format the whole notebook.
Functions

Functions are key to Mathematica: functions receive information or data, process it, and return a result.

Functions are called by their name, usually composed of complete English words describing what the function does, with no spaces and first letters capitalized.

Function names are followed by square brackets, in which one or more arguments is entered:

\[ \text{FunctionName[argument]} \]

For example, the \text{ListPlot[]} function takes a matrix of x,y values as its argument:

\[ \text{ListPlot[\{\{1,2\},\{3,4\}\}}] \]

Mathematica’s help files give descriptions and examples of every function.
Options for functions

Many functions have options that are entered as arguments

Options usually have the format 
`OptionName -> Value`

Find options with `Options[FunctionName]` or in Documentation Center
Variables

Variables are also key to *Mathematica*, allowing you to store information.

Variables do not have brackets or options.

You create variables, giving them a name and putting something into them.

Here a variable called `data` is used to store a number, a sequence of numbers, the natural log of a sequence of numbers, and data imported from an Excel file. A variable called `mygraph` is used to store a graphic.

You can retrieve what is inside a variable by executing it (the graph is displayed again by executing `mygraph`).
Parts of variables

When a variable has more than one item stored, you can get specific parts using double square brackets after the variable name.

*data* returns all the items in *data*

*data[[1]]* returns only the first item in *data*

*data[[1;;3]]* returns items 1 to 3

For more examples look at the Documentation Center under the function *Part[]* and under the tutorial *GettingPiecesOfLists*.
Lists, Matrices, and other Multidimensional data

You will often work with “lists”, which is Mathematica’s term for any group of several items

Some lists have only one element (scalar), some have a long row of elements (vector), some have columns and rows of data (matrix or array)

You can get columns, rows, or elements from the list using the double square bracket system

See Documentation Center under:

1. ListsOverview
2. HandlingArraysOfData

In[12]:= data = 1
Out[12]= 1

In[26]:= data = {10, 20, 30, 40, 50, 60}
Out[26]= {10, 20, 30, 40, 50, 60}

In[27]:= data = {{1, 2}, {1, 3}, {3, 1}}
Out[27]= {{1, 2}, {1, 3}, {3, 1}}

In[28]:= data = {{{1, 2}, {1, 3}, {3, 1}}, {{1, 2}, {1, 3}, {3, 1}}}
Out[28]= {{{1, 2}, {1, 3}, {3, 1}}, {{1, 2}, {1, 3}, {3, 1}}}

In[30]:= data[[1, 2 ;; 3]]
Out[30]= {{1, 3}, {3, 1}}
Special formatting tags

You can control the display of output in many ways by putting special tags at the end of a line of input.

semicolon (;) prevents output from being displayed

//N forces numbers to be displayed in decimal form

//MatrixForm displays tables of data in rows and columns.
Importing and exporting data

Mathematica has an extensive range of file types that can be imported and exported: text files, Excel files, Word files, PDFs, Illustrator, JPEG, etc.

```
Import[FilePath]
Export[FilePath, "type"]
```

Note the helpful file path chooser found on the Insert menu
Simple graphics

ListPlot[]

Plot[]

Histogram[]

BarChart[]
**Loops:** programming structure for repeating things

Use `Table[]`, `Map[]`, or `Do[]` to carry out repeated tasks

`Table[ lines to be repeated , \{ iterator \}]`

where the lines to be repeated consist of other Mathematica functions or lists of functions separated by semicolons

`iterator` is a special construction that creates a temporary counting variable and specifies number of times to repeat

**Simple:** `{10}` (repeats 10 times)

**With variable:** `{x, 10}` (repeats while incrementing x from 1 to 10 in steps of 1)

**Full:** `{x, 1, 10, 1}` (repeats while incrementing x from 1 to 10 in steps of 1)

**Full:** `{x, 10, 2, -2}` (repeats while incrementing x backward from 10 to 2 in steps of 2)
Conditional statements

Is equal?  ==
Is unequal?  !=
Greater than?  >
Less than?  <
And   &&
Or  ||

If[ statement is true,  then this,  or else this ]

myage = 65.5;
If[ myage > 50, Print[“my age is older”], Print[“my age is not older”] ]

If[ myage > 55 && myage < 65, Print[“my age is in the bin”], Print[“my age is outside the bin”] ]
Working with Strings

Strings are entities of characters, as opposed to numbers. You can manipulate strings in Mathematica as well as numbers. For example:

mytext = “Species”;

You can combine strings by joining them with the StringJoin[] function or <> (which do the same thing):

\begin{verbatim}
In[16]:= StringJoin[mytext, " Name"]
Out[16]= Species Name
\end{verbatim}

\begin{verbatim}
In[17]:= mytext <> " Name"
Out[17]= Species Name
\end{verbatim}

You can create a list of labels using Table[] and ToString[], the latter of which converts numbers to strings so they can be joined to other strings:

\begin{verbatim}
In[18]:= Table[mytext <> " " <> ToString[x], {x, 5}]
Out[18]= {Species 1, Species 2, Species 3, Species 4, Species 5}
\end{verbatim}
Random numbers

Mathematica has many functions for generating random numbers.

(* Random real number from 0 to 1 *)

\begin{verbatim}
In[4]:= RandomReal[]
Out[4]= 0.9513
\end{verbatim}

(* random real number from 100 to 1000 *)

\begin{verbatim}
In[5]:= RandomReal[{100, 1000}]
Out[5]= 505.785
\end{verbatim}

(* 10 random real numbers from 100 to 1000 *)

\begin{verbatim}
In[10]:= RandomReal[{100, 1000}, 10]
\end{verbatim}

(* Random number drawn from a normal distribution with a mean of 10 and standard deviation of 100 *)

\begin{verbatim}
In[11]:= Random[NormalDistribution[10, 100]]
\end{verbatim}

(* 10 pairs of random numbers between 0 and 1 *)

\begin{verbatim}
In[12]:= Table[RandomReal[{0, 1}, 2], {10}]
Out[12]= {{0.0245927, 0.630284}, {0.260035, 0.591502}, {0.38211, 0.146923},
    {0.891077, 0.0315945}, {0.75184, 0.567132}, {0.553506, 0.443656},
    {0.614652, 0.300159}, {0.791076, 0.0654448}, {0.19977, 0.272843}, {0.291167, 0.958036}}
\end{verbatim}
Defining your own function

You can create your own customized functions to perform operations that you use a lot.

The syntax uses “:=” to define the operation of the function.

The input parameters are defined as variables with an underscore after them.

The Module function shields the variables used in the custom function from the rest of the notebook (it keeps them from clashing).

Custom functions usually end with Return, which is a function that returns something to the user in response to the input parameters.

This example takes two numbers as input, adds them together and multiplies them by 10, and stores the result in the temporary internal variable \( j \). The value is returned to the user at the end of the function.