Ph3 Mathematica Homework: Week 6

Eric D. Black
California Institute of Technology
v1.2

Now that we've covered techniques specific to data analysis, we will branch out into some more general topics. This week all the exercises are at the end of the handout, so if you already know these commands you can just skip to the end and do them.

1 Formatting

Up to now we have been writing our expressions Fortran-style, like this

```
(-b + Sqrt[b^2 - 4*a*c])/(2*a)
```

Mathematica also allows you to format equations so they look like what you'd write on a page, and it's no more difficult to enter than the Fortran-style notation.

$$\frac{-b+\sqrt{b^2-4\,a\,c}}{2\,a}$$

There are pallets you can call up to help type these in, but once you get the hang of it it's easier to use the keyboard. Here are a few keyboard shortcuts that are useful to know. (As usual, `<Ctrl>` means hold down the control key while pressing another key, using it as you would a shift key. `<Esc>` means press and release the escape key.)

1. Multiplication - `*` or `<Spacebar>`

2. Fractions - <Crtl> /
3. Exponents - <Ctrl> ^
4. Subscripts - <Ctrl> _
5. Square-roots - <Ctrl> 2
6. Greek letters - <Esc> alpha <Esc> , etc. Upper case is made by capitalizing the name of the letter, *e.g.* <Esc> Omega <Esc>. Note that π is reserved.
7. Imaginary number i ($\sqrt{-1}$) - I or <Esc> ii <Esc>
8. Euler's number e - <Esc> ee <Esc>
9. Integral symbol ($\int$) - <Esc> int <Esc>
10. Lower limit of a definite integral - <Ctrl> _
11. Upper limit of a definite integral - <Ctrl> %
12. Integral dx notation - <Esc> dd <Esc> x
13. Summation symbol (Σ) - <Esc> sum <Esc>. Limits are entered the same as in integration, except that the lower one should contain the indexing variable as well as its first value, as in $\Sigma_{n=1}^{100}$.

2 Commenting

You can comment a Mathematica notebook by designating certain cells as text-only.

With the cursor inside an empty cell hold down the Command key, and type 7 (<Cmd> 7). This will make that particular cell a comment cell (text-only), and you can type any comments you want. Text-only cells support all of the regular formatting commands, so you can make equations, etc.

You can comment individual lines or bits of text inside a cell by enclosing them in paretheses and astersks, like so.

```
(* This is a comment. *)
```

3 Calculus

You may suspect by now (or already know) that Mathematica can do basic mathematical manipulation for you. The basic format for definite integrals is

```
Integrate[f[x], {x, min, max}]
```

If you leave out the bounds you will get an indefinite integral.

```
Integrate[f[x], x]
```

You can also use the integral symbol as described above, and Mathematica will understand and give you the appropriate answer.

The derivative of $f(x)$ with respect to x can be written two ways.

```
D[f[x], x]
```

or

```
f'[x]
```

The second form has the feature that it can be used to evaluate derivatives at a particular point. For example,

```
f''[0.7]
```

means the second derivative of f with respect to x, evaluated at $x = 0.7$, or

$$\left.\frac{\mathrm{d}^2 f}{\mathrm{d}x^2}\right|_{x=0.7}$$

You can solve differential equations.

```
In[32]:= DSolve[y'[x] / y[x] == a, y[x], x]
Out[32]= {{y[x] → e^(a x) C[1]}}

In[33]:= DSolve[y''[x] + (a - 2 * q * Cos[2 * x]) * y[x] == 0, y[x], x]
Out[33]= {{y[x] → C[1] MathieuC[a, q, x] + C[2] MathieuS[a, q, x]}}
```

One point needs to be made here. Notice how I have used double equals signs (`==`) in the differential equations instead of a single equals (`=`). The two expressions mean very different things. A single equals is really a command to Mathematica. It says, for example, "set the value of the variable `a` to 2.0."

```
a = 2.0
```

A double equals sign, however, is a logical statement, the starting point for a chain of reasoning or simply a test.

Mathematica will take limits for you too. Use the `Limit` command. Syntax is

```
Limit[1/x^2, x -> 0]
```

And yes, it will return infinity (∞) if that is the correct answer.

4 Algebra

Mathematica will also do algebra for you.

```
In[22]:= Factor[1 - x^3]
Out[22]= -(-1 + x) (1 + x + x^2)

In[30]:= Solve[5 x + 3 == 13]
Out[30]= {{x → 2}}
```

(Notice the double equals sign in an equation, as opposed to a single equals for an assignment.) This is honestly of only limited utility in my experience. It's a cute trick, but more often than not its just easier to do algebra yourself. Having Mathematica do a series expansion, on the other hand, can be useful. The syntax for that is

```
Series[f[x], {x, x_0, n}]
```

where `f[x]` is the function you want expanded, `x` is the independent variable, `x_0` is the point you want the series expanded about, and `n` is the number of terms you want.

5 Arithmetic

You have seen how Mathematica can be used as a calculator and how to get numerical results using the `N` function. It will probably not surprise you to learn that it complex numbers are supported as well.

```
z = a + I*b
```

Some functions you should be aware of that relate to numbers and arithmetic are

1. `N[f, n]` - Evaluate a numerical result (approximation) of an expression `f` to `n` significant figures. The argument `n` is optional.
2. `Abs[z]` - Absolute value of the number `z`. This is the magnitude when `z` is complex.
3. `Arg[z]` - Argument (complex phase) of the number `z`.
4. `PrimeQ[n]` - Test an integer `n` to see if it's prime.
5. `FactorInteger[n]` - Factor the integer `n` into its prime components.

6 Numerical methods

Some very useful commands for finding numerical solutions are,

1. `NIntegrate[f[x], {x, min, max}]` - Evaluate the definite integral of `f[x]` over `x` from `x=min` to `x=max` numerically.
2. `NDSolve[eqns, y, {x, min, max}]` - Numerical solution to differential equations `eqns` in `y` over the range `x=min` to `x=max`. The equations `eqns` must include boundary conditions, *e.g.* for `eqns` you could write `{y''[x]==y[x]*Cos[x+y[x]], y[0]==1}`.
3. `NSolve[expr, vars]` - Numerically find solutions to an expression `expr` (*e.g.* an equation, system of equations, or inequalities) for the variables `vars`. Restrict to only real solutions with the additional option `Reals`.
4. `FindRoot[f, {x, x0}]` - Similar to `NSolve`. Finds roots (zeros) of an expression `f`, starting with the initial guess `x=x0`.

7 Procedural programming

Those of you who are used to procedural programming will almost certainly have asked by now, "How do I code for loops in Mathematica, or if-then statements?" Well, `Table` can handle most of your looping requirements, and you've already seen that. Or you could use the `Do` command, which does just what you'd expect.

```
Do[expr, {i, min, max}]
```

This evaluates an expression `expr` sequentially over the index `i` from `min` to `max`. Decision making is handled similarly.

```
If[expr, then, else]
```

This evaluates `expr`. If it's true, then it executes whatever is in `then`. If not, then it executes `else`. The expression `else` is optional, but it's good programming practice to include it. Here's an example.

```
If[y != 0, q = x/y, Print["Can't divide by zero!"]]
```

Note the use of logical tests like `!=` or `==`, which we covered last week, in `expr`. These expressions can be combined as well, and two of the most useful notations for doing that are,

1. `&&` - AND
2. `||` - OR

There are expressions for many other logical tests, such as XOR, NOR, XAND, NAND, NOT, etc. For more information on those, see the documentation (`http://reference.wolfram.com/language/`).

You can, of course, nest these to produce a traditional DO loop with decision statement in one line.

```
Do[If[expr, then, else], {i, min, max}]
```

8 Exercises

Exercise 1: Type the following lines into Mathematica, and execute them.

1. `1 = 2`
2. `1 == 2`
3. `1 != 2`
4. `1 < 2`
5. `1 <= 2`
6. `1 > 2`

You see how Mathematica uses these expressions? The logical tests are very useful in procedural programming.

Exercise 2: Check your phone number to see if it is prime, with and without the area code. What are the chances it would be? (If you'd rather keep your phone number private, you may make up a random seven-digit number and area code and use that.) *Hint:* A reasonable estimate of the number of primes up to a given integer n is $n/\ln(n)$. A more accurate number is given by the prime counting function $\Pi(n)$, which in Mathematica is called `PrimePi[n]`.

Exercise 3: Write a short program that tests a number to see if it's prime. If it is, have it print out a short message. If it's not, have Mathematica factor it and print out its prime factorization. It would be good practice to test the number to see if it's an integer before doing anything else, and print out an error message if it's not an integer.