Macaulay2

An Introduction

Matthew Weaver

Purdue University

January 2019

So what is it?

- Macaulay2 is a computer algebra system for work in algebraic geometry and commutative algebra.
- Can be used to compute certain invariants and run examples
- Some built-in functions include:
  - Free resolutions, Betti tables, multiplicities etc.
  - Dimension, codimension, depth, etc.
  - Able to characterize important algebras: symmetric algebra, Rees algebra, etc.

Getting started

- Macaulay2 is built into the MATH office computers under "programming"
- Can be downloaded onto any Linux machine and started from the command line.
- Can be run via a remote desktop to the department servers (preferred method for today)
- Can also be used online through Cornell's emulator, Habanero
- Regardless you should arrive at a prompt that looks like this:

Inputs and Outputs

- Your input lines are labelled by “i” and your outputs by “o” followed by their number.
- Enter your command and hit “Enter” to run an input
- You may receive multiple outputs, such as a new label or a description
- Most of the common commands are already recognized by Macaulay2, but you might need to load additional packages depending on your needs
- A full list of commands can be found on the Macaulay2 website in the index or documentation section

Equalities and inequalities

Most of the time we care about relating numeric invariants and equality or containments of certain objects:

- Macaulay2 has **Four** types of equalities:
  - "=" : Used to assign a name or label to an object
  - ":=" : Used to assign a name to a new local object
  - "==": Used to verify if two objects are equal in the normal sense
  - "===": Used to check if two objects are **identically** equal (be careful with this one)
- "<" and ">" still represent less than / greater than
- Use "<=" and ">=" to denote less than or equal / greater than or equal

Writing your own commands and functions

- Start off by giving your command a name followed by "="
- Next identify your inputs (use parentheses if multiple)
- Type "$-$ $>$" and then what you want to do with your inputs
- Be careful! Once you hit enter and drop down a line (like if there's an open parenthesis and your program can't run) you will not be able to go back up
- Useful to write your commands in another program (notepad, Overleaf, etc.) and then copy and paste into Macaulay2

Exercise: S-pairs

Definition

Recall that for nonzero polynomials f and g of $R = K[x_1, \ldots, x_n]$ and monomial ordering $<$, the S-Pair of f and g with respect to this ordering is given by:

$$S(f,g) = \frac{\operatorname{lcm}\big(\operatorname{in}_<(f), \operatorname{in}_<(g)\big)}{c_f \operatorname{in}_<(f)} f - \frac{\operatorname{lcm}\big(\operatorname{in}_<(f), \operatorname{in}_<(g)\big)}{c_g \operatorname{in}_<(g)} g$$

where c_f and c_g represent the coefficients of the initial terms.

Exercise

Write a function that computes the S-pair of two polynomials.

For Loops

- Used to parse through and indexed list and complete an operation at each step
- Should look like:

For "index" from **a** to **b** when **c** do **d**

- The first two entries **a** and **b** represent the span of the list you wish to go through
- The third entry **c** is any condition or restriction you wish to impose
- The last entry **d** is the actual operation you wish to perform
- Not all of these are necessary however!

While Loops

- Used to repeatedly execute a command until specified conditions are no longer met
- should look like:

$$\text{while } \_\mathbf{c}\_ \text{ do } \_\mathbf{d}\_$$

- The first entry **c** is the condition you impose and the second entry **d** is the command you want executed
- May need to specify a starting value for your object if you have not declared one already
- Be careful! Since this runs continually you must ensure the program terminates
- May also need to ensure your object changes after each iteration so that the necessary condition is eventually false

Conditional Statements

- Should look like

  if **a** then **b**

- If an input is entered that doesn't fall into any of your cases, the null output will be returned.
- Cover specified other cases with

  else if **c** then **d**

- End with

  else **e**

  when you've exhausted all other possibilities or you don't care about the alternative cases

Conditional Statements

- Should look like

  if **a** then **b**

- If an input is entered that doesn't fall into any of your cases, the null output will be returned.
- Cover specified other cases with

  else if **c** then **d**

- End with

  else **e**

  when you've exhausted all other possibilities or you don't care about the alternative cases

Exercise

Write a function to test whether a ring is Cohen-Macaulay or not.

Calling commands

- You can use any function you have created by calling it's name
- Previous functions can be used to define new functions
- **Tip:** Write a bunch of smaller one-objective commands first and call them when necessary

Calling commands

- You can use any function you have created by calling it's name
- Previous functions can be used to define new functions
- **Tip:** Write a bunch of smaller one-objective commands first and call them when necessary

Exercise

Write a function to test whether a ring is Gorenstein or not.

Calling commands

- You can use any function you have created by calling it's name
- Previous functions can be used to define new functions
- **Tip:** Write a bunch of smaller one-objective commands first and call them when necessary

Exercise

Write a function to test whether a ring is Gorenstein or not.

Definition

Recall that for a local ring $(R, \mathfrak{m}, k)$ and nonzero finite R-module M, the **type** of M is

$$\underline{r(M)} = \dim_k \operatorname{Ext}^t_R(k, M)$$

where $t = \operatorname{depth} M$.

Tips when Writing Functions

- Comments can be made by typing "- -"
- If you are using multiple mathematical objects, make sure they are compatible! Useful commands to help with this are
  - "module(__)": turns a ring or ideal into a module
  - "sub(__ , __)": Allows one to consider one object in a different setting (example: extending ideals)
  - When considering a module over two different rings, a tensor product "**" can help distinguish which structure to use
- If you wish to stop a command that is still running (taking too long or an infinite loop) enter Ctrl+C (not the copy shortcut)

Tips for Writing Functions (cont.)

- Some commands can be time consuming and may never finish
- Be smarter than the computer! A lot of invariants can be computed faster using known formulas and theorems: Auslander-Buchsbaum, Dimension formulas, etc.
- A brief description of how a built-in command works can be found on the Macaulay2 webpage
- Just because your code runs doesn't mean it's correct! Test it on known results to be sure!