Reproducible Genomic Project

and a bit of computational thinking

Caleb Kibet

September 16, 2019

Introduce yourself and answer the question...

What is you exposure to Linux and Computing?

- I have used Linux terminal
- I have used one or more programming language (R, Python, etc)
- I know about algorithms
- These are all new to me, but I am ready to learn

Why should you learn to about Computing?

Alan Perlis

- One of the founders of computer science
- Argued in 1961 that Computer Science should be part of a liberal education: **Everyone should learn to program.**
  - Perhaps computing is more critical to a liberal education than Calculus
  - Calculus is about *rates*, and that's important to many.
  - Computer science is about *process*, and that's important to **everyone**.

Programming is about Communicating Process

An **algorithm** is a sequence of steps for solving a specific problem given its input data and the expected output data. An algorithm transforms the input to output.

A **program** is the most concise statement possible to communicate a process in a specific programming language.

..a set of instructions that instructs a computer to carry out certain operations

- translate DNA
- predict a motif
- perform statistical analysis
- align sequences

A Computer is Stupid, but this is about to change

"A computer is a stupid machine with the ability to do incredibly smart things, while computer programmers are smart people with the ability to do incredibly stupid things."
– Bill Bryson,

You have to tell it everything it needs to do, and how to do them...

An algorithm is a mechanical procedure that is guaranteed to eventually finish.

Algorithm to make coffee

For example, here is a procedure for making coffee, adapted from the directions that come with a major coffeemaker:

1. Lift and open the coffeemaker lid.
2. Place a basket-type filter into the filter basket.
3. Add the desired amount of coffee and shake to level the coffee.
4. Fill the decanter with cold, fresh water to the desired capacity.
5. Pour the water into the water reservoir.
6. Close the lid.
7. Place the empty decanter on the warming plate.
8. Press the ON button.

Describe a simple algorithm

Cook Ugali?

Programming language

Programming is the act of writing instructions that make the computer to solve a problem. A **programming language** is the tool that is used to implement an algorithm.

Figure: Some programming languages

What is a problem?

They are of two types:

1. A problem with algorithmic solution

- They have a clearly defined sequence of steps that would give the desired solution eg baking, adding two numbers...
- the sequence of steps or recipe for arriving at the solution is called the **algorithm**.

2. A problem with heuristic solution

- Solutions emerge largely from the process of trial and error based on knowledge and experience (learning) eg, winning tennis,
- **Machine Learning** is commonly used to solve such problems

Problem-solving steps

1. Defining the problem requirements (R)
   - Clearly define the problem in words, stating the input and output data as well as the processing logic. Requires familiarity with the problem environment and needs
2. Identifying Problem Components (C)
   - From the problem definition, identify the list of problem inputs, outputs, constraints and relationships between input and output data expressed in coherent formulas.
3. Possibly break problem solution into small modules (M)
   - Skip this step when dealing with small problems
4. 4. Design the Algorithm to solve the problem (A)
   - Chose the best among many alternative ways for solving the problem
   - Define algorithmic solution for all modules
5. Implementation and Coding (C)
   - Translate the algorithm into a program using a programming language
6. Test the program to ensure that it gives the expected output (R)

You're a scientist first, not a programmer

Remember you are a **scientist** *and the* **quality of your research** *is what is important,* **not how pretty** *your source code looks.*

Perfectly written, extensively documented, elegant code that gets the answer wrong is not as useful as a basic script that gets it right.

Computational Thinking

Computational thinking is reformulating a seemingly difficult problem into one we know how to solve, perhaps by reduction, embedding, transformation, or simulation.

- **decomposition**: breaking down a complex problem or system into smaller, more manageable parts
- **pattern recognition**: looking for similarities among and within problems
- **abstraction**: focusing on the important information only, ignoring irrelevant detail
- **algorithms**: developing a step-by-step solution to the problem, or the rules to follow to solve the problem

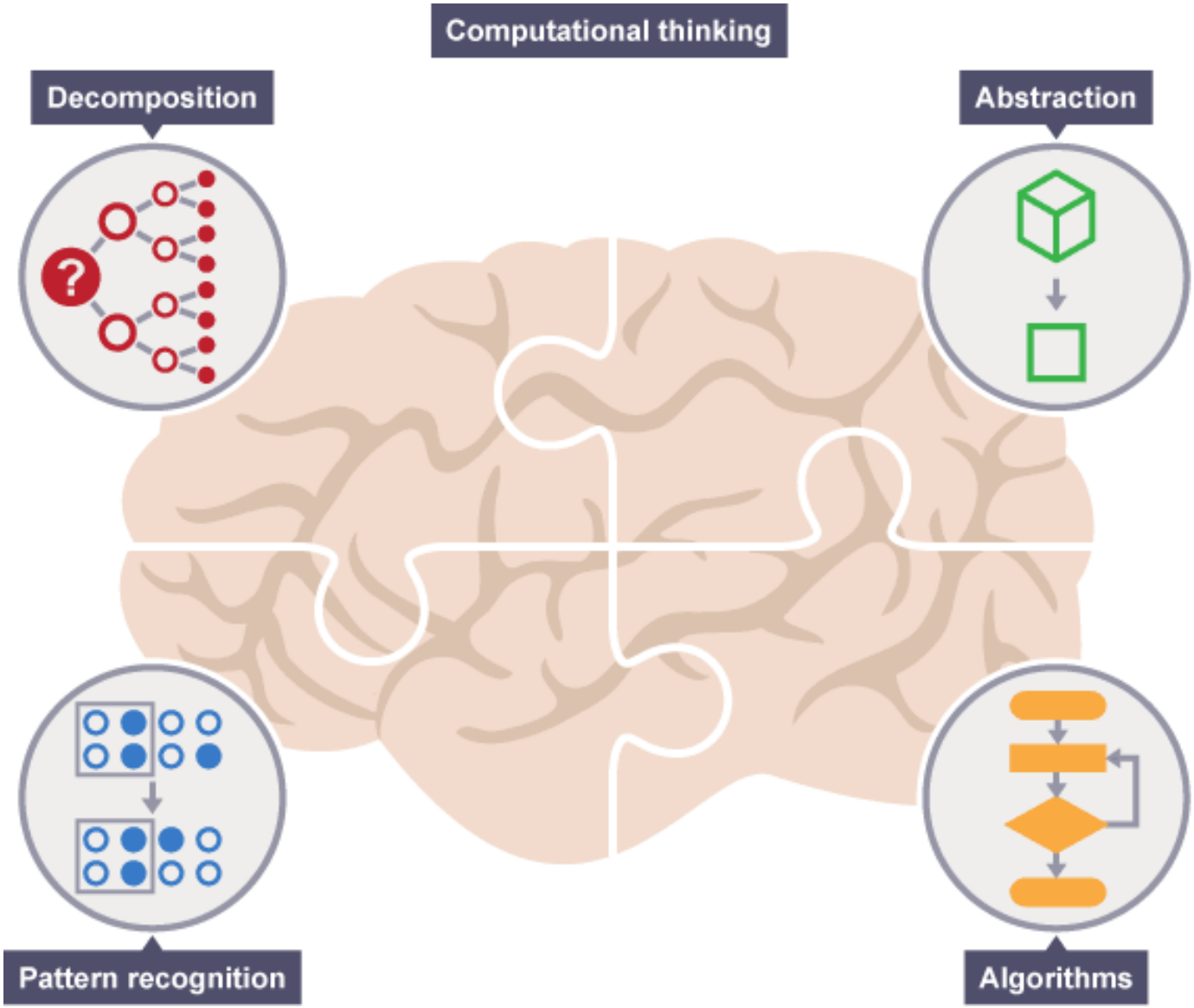

Figure: Computational Thinking. Source: https://www.bbc.com/bitesize/guides/zp92mp3/revision/1

Reproducible Research and Data Analysis

Reproducibility should be core to your Bioinformatics data analysis

Reproducibility means that research data and code are made available so that others are able to reach the same results as are claimed in scientific outputs.

1. Formulating a hypothesis
2. Designing the study
3. Running the study and collecting the data
4. Analysing the data
5. Reporting the study

Each of these steps should be clearly reported for transparency and reproducibility.

Tips for Reproducible Research

Document everything

- Everything is a (text) file.
- All files should be human readable.
- Explicitly tie your files together.
- Have a plan to organize, store, and make your files available.
- Report your research transparently

Tips for Reproducible Research

Document everything

- Everything is a (text) file.
- All files should be human readable.
- Explicitly tie your files together.
- Have a plan to organize, store, and make your files available.
- Report your research transparently

Keep track of things

- Use version Control
- Use proper documentation: README
- Use Literate programming: RMarkdown, Jupyter Notebooks

Tips for Reproducible Research

Share and license your work

- Data: Adhere to FAIR data principles
- Software: Github and other repos

Project Folders

- Choose a file structure that works for you
- Use relative paths when possible and organize your files: Makes paths less dependent on particular File or System structure.
- Avoid putting spaces in your file and directory names
- Include a README that describes the purpose and structure of your project

Tips for Reproducible Research

Before you start

- Create project within a folder in your computer
- Create folder for your code
- Create folder for Data
  - Raw: Downloaded or gathered from the field
  - Derived: processed through your analysis
- Create a folder for figures generated from your analysis NB: Ensure separation of information

What makes a good programmer?

- Logical thinking
- Creative thinking
- Problem solving skills
- The ability to "think outside of the box"

Be suspicious and trust nobody, especially Bioinformatics Tools...

Computational analysis, especially with large data, **will always give you some results** (significant p-value).

Treat results **with great suspicion**, and carry out further tests to determine whether the results can be explained by experimental error or bias

Chose a tool based on your question, not for the sake of it

- You want to sequence, why?
- You want to align, why?
- You want a tree, why?

Errors are opportunities to learn, embrace them

You will definitely have errors in your code. We all do. Learning to interpret the error message, and identify the problem, is an essential skill to acquire from the onset.

What's Next

- Learn Linux and the command line
- Learn Python
- Work on a Python Project

Training Format

We'll mostly use a combination of lectures and live coding.

References

To go deeper into this topic, read:

1. : Carey MA, Papin JA (2018) Ten simple rules for biologists learning to program. PLoS Comput Biol 14(1): e1005871. https://doi.org/10.1371/journal.pcbi.1005871
2. Loman, N., and Watson, M. (2013). So you want to be a computational biologist? Nat. Biotechnol. 31, 996.introduction