Ten Simple Rules for Making Research Software More Robust: The Checklist

Morgan Taschuk[1,‡*], Greg Wilson[2,‡]
1 Ontario Institute for Cancer Research / morgan.taschuk@oicr.on.ca
2 Software Carpentry Foundation / gvwilson@software-carpentry.org

‡ These authors contributed equally to this work.
* Corresponding author.

- ☐ **Have a README**
  - ☐ Explain what the software does
  - ☐ List required dependencies
  - ☐ Provide compile/installation instructions
  - ☐ List input and output files
  - ☐ State attributions and licensing
- ☐ **Print usage information from the command line**
  - ☐ Include:
    - ☐ the syntax for running the program in GNU/POSIX format
    - ☐ a one line description
    - ☐ the most commonly used arguments, a description of each, and the default values
      - ☐ Where to find more information
  - ☐ Print to standard output
  - ☐ Exit with an appropriate exit code
- ☐ **Version your releases**
  - ☐ Increment your version number every time you release your software to other people
  - ☐ Print when supplying `--version` or `-v` on the command line
  - ☐ Include version number in output
  - ☐ Deposit releases in a stable location so they are available in perpetuity
- ☐ **Reuse software (within reason)**
  - ☐ Make sure that you really need the auxiliary program
  - ☐ Check for dependent software and version early in execution
  - ☐ Use native functions for starting other processes
- ☐ **Use a build utilty and package manager**
  - ☐ Document *all* dependencies, preferably in a machine-readable form
  - ☐ Avoid depending on scripts and tools which are not available as packages
- ☐ **Do not require root or other special privileges**
  - ☐ Allow packages to be installed in an arbitrary location
  - ☐ Ask another person to try and build your software
- ☐ **Eliminate hard-coded paths**
  - ☐ Set the names and locations of input and output files as command-line parameters
  - ☐ Do not require users to navigate to a particular directory to do their work
- ☐ **Allow configuration of all useful parameters from the command line.**
  - ☐ Choose reasonable defaults where they exist
  - ☐ Set no defaults at all when there aren't any reasonable ones
  - ☐ Echo all parameters and software versions to standard out or a log file alongside the results
  - ☐ Check that all input values are in a reasonable range near startup

- ☐ **Include a small test set that can be run to ensure the software is actually working.**
  - ☐ Tests are easy to find and run
  - ☐ Test results are easy to interpret
- ☐ **Produce identical results when given identical inputs**
  - ☐ For randomized algorithms: allow the user to optionally provide the seed as an input parameter; or
  - ☐ Make sure the acceptable tolerance is known and detailed in documentation and in the tests

Bonus points

- ☐ **Conform to command-line conventions** [**1**]
- ☐ **Write high quality documentation** [**2**]

References

1. Seemann T. Ten recommendations for creating usable bioinformatics command line software. GigaScience. 2013;2(1):15. doi:10.1186/2047-217X-2-15.

2. Karimzadeh M, Hoffman MM. Creating great documentation for bioinformatics software; 2016. `http://hdl.handle.net/1807/73111`.