When and How to Introduce Students to Free and Open-Source Software

Inaugural GVSU Technology Summit

Quinton Quagliano, M.S., C.S.P. Psychology Department

Table of Contents

1 Introduction

1.1 Disclosures and Disclaimers

- No AI-based tools have been used in the creation and writing of this presentation
- I have no disclosures or conflicts-of-interests related to this presentation or the software described in it
- I am not a software engineer, computer scientist, or other technology-oriented professional by training - but I am an enthusiast, hobbyist, and advocate!

1.2 Learning Objectives

- Listeners should appreciate *why* we, and our students, should pay attention to how software is published and priced (Sections: Motivation and Purpose)
- Listeners should understand the vocabulary used to describe pricing models and source code availability in software (Section: Vocabulary of Software Availability and Pricing)
- Listeners should be able to identify reasonable open-source alternatives to popular and proprietary tools used in their area of work (Section: How to Find Alternative and Open-source Tools)
- Listeners should learn about some practical implementation examples to bring more diverse software to students in class (Section: Examples of Integration)
- Listeners should recognize both the advantages and disadvantages of adopting open-source alternatives into instruction (Section: Throughout Examples of Integration)

1.3 Motivation

- The four (hyperbolic) “Evil” Es of software
  - Software is **everywhere** - it’s always all around us
  - Software is **essential** - it’s a common requirement of navigating the world
  - Software is **elaborate** - but it doesn’t look it!
  - Software is **expensive** - and keeps getting more so!
- I want my students able to responsibly navigate these “Evil Es” during and after college, and not feel lost when they encounter new technology

1.4 Purpose

- Support the liberal arts mission of creating well-(tech)-rounded students

  - Expose students to more alternative tools
  - Help students see the similarities, differences, and quirks of each tool
  - Ensure that when students encounter new software they can adapt easier
- Support projects and software that are free in a time of increasing prices
  - Push back against reliance upon subscription-based and more “locked-down” tools
  - Show students how to build their portfolio and skill set without incurring additional financial burden
  - Make sure that college education does not rely upon the solvency of software businesses and startups - students should leave with the flexibility necessary to outlast individual products!

> **Important**
>
> This is **not** an attempt to insist upon **only** using open-source software

2 Vocabulary of Software Availability and Pricing

2.1 Basic Software Terms

- **Developer(s):** The creator(s) and maintainer(s) of a certain piece of software, may be a somewhat unorganized group of like-minded people or a centralized corporation
- **End User:** A person who uses a certain piece of software for personal or business use, i.e., You and I
- **License:** Some document or text shared with software, written by the developer(s), that dictates how a certain software, and its source code, can or cannot be used, modified, and shared
  - When used correctly, this is legally binding and ignoring license terms can be grounds for lawsuits
  - E.g., MIT License, CC, specific licenses written for proprietary software
- **End User License Agreement (EULA):** An agreement signed by a end user of a specific software that recognizes and agrees to the software license terms set by the developer, and the penalties associated with violating the license
  - I.e., the long set of documents and text we agree to when we sign up for a new service
- **Source code:** the actual code that underlies a certain software or program
  - This is *not* necessarily the files that “ship” with the software
  - This is used to build and modify the program, and in “closed-source” software, is only available to the developer(s)

2.2 Terms of Ownership and Use

2.2.1 Non-free Software

- Software that is restricted with limited or no access to source code. Contrary to the name, it does not necessarily have a cost to use
  - **Proprietary:** A type of non-free software; Something that is owned by developer(s) via copyright and intellectual property laws, and end users are not allowed to make limitless modification to the program or share without authorization
    - E.g., Microsoft 365 (Word, Powerpoint, etc.) - owned by Microsoft, cannot be modified or shared without permission from copyright holder
  - **Software as a Service (SaaS):** Proprietary, non-free software priced as a continuous ongoing subscription model, rather than charging a one-time access fee
    - E.g., Adobe Creative Cloud, likely anything that says: “Contact our Sales Team”

2.2.2 Free and Open Source Software (FOSS)

- Software that is *intentionally* not priced, and able to be adopted, modified, shared, and used without cost - with only some restrictions per the specific license of the software
  - The “open source” part of this, refers to the source code being open and available, i.e., not hidden from the public
  - A subtype of this is **Free, libre, and open source (FLOSS) or copyleft**, which focuses on prohibiting the use of the software to create non-free software
- For the purpose of this presentation, I’ll be prioritizing talking about options for using and embracing **FOSS** software

> **Important**
>
> There’s a lot of different ways to describe software price models, and sometimes a lot of grey area too - so be discerning when looking at options!

3 How to Find Alternative and Open-source Tools

3.1 Strategy

1. Identify which programs and software are “industry-standard” in your field of study/teaching and which programs you would normally suggest/have students use for assign-

ments in the classes you instruct

2. For each of those identified programs, list what *core* features they have that your students should have in any alternative program, i.e., what parts of those programs are necessary to achieve the desired learning objectives and complete assignments
3. Use these websites to help find similar programs to those you already use:
   - AlternativeTo
   - SourceForge
   - Google - search “Open-source alternative to [insert proprietary program name here]”
   - AI chats and tools can help too - just be wary of hallucinations!
   - Surprisingly, internet forums like Reddit can also be useful to check!
4. Try out alternatives and see if they match your needs - you want to make sure that it works well enough so you would feel comfortable with your students using it!

3.2 Things to Look for in Software

- Here is some general guidance of things to look when considering whether software seems to be legit and well-supported
- Use the Vocabulary of Software Availability and Pricing to help cut through the marketing, jargon, and fluff!

3.2.1 Green Flags

- Supported by community involvement/discussion and code contributions
- Recent activity, updates, and contributions
- Source code publicly available and easily find-able
- Supported by voluntary donations ONLY, with no features gated by default
- Backed by non-profit

3.2.2 Red Flags

- Prominent advertising for subscription tiers, “membership”, or overbearing donation suggestions, “Contact sales”, “free trial”
- Difficult to find source code
- Un-maintained, out-of-date code repositories
- Low or nonexistent community discussion or involvement

3.3 Common Alternative Examples

- I don’t guarantee that the following suggestions are all of the same skill level, some may require some time investment to re-learn

- Microsoft / Google Suite for *offline* document, sheets, and slideshow preparation → LibreOffice, Apache OpenOffice
  - For this presentation, I'm using RevealJS, written with Quarto, as an alternative to using MS Powerpoint
- Github for code sharing, version control, other code actions → Codeberg, Gitlab, Gitea
- Chatgpt / Gemini / Other AI chats for generic LLM use → Ollama
  - I greatly prefer this option for privacy reasons as well
- SPSS / SAS / STATA for statistics → R, Python
- *Some* Adobe programs like Photoshop and Illustrator → GIMP, Inkscape
- Panopto for video / screen recording → Open Broadcaster Software (OBS)
  - I use this for recording all of my lectures

> **Important**
>
> Trying to find a real, legit, useful, and feature-rich alternative can be a minefield! Take your time to parse out the best solutions, but there isn't always one!

4 Examples of Integration

4.1 General Advice

- Take the below with a grain-of-salt; you may have to do a little or a lot of modifications to these to make them make sense for your classes
- Realistically, not all classes may benefit from addition of software related/assisted instruction - add in relevant activities as appropriate to your learning objectives

4.2 Option 1: A Longer Assignment with a Choice of Two (or More) Ways

4.2.1 Steps

1. Think of an assignment you already give to your class in which you recommend/suggest/require a proprietary software
2. Rewrite assignment instruction to allow for use of the proprietary *or* the open-source software alternative (make sure you vet the open source alternative first and make

sure it is reasonably close in difficulty!), but keep outcome expectations / grading standards roughly the same

3. If one of the choices presents a greater learner curve or challenge, consider offering some *marginal* extra points for use of the more difficult tool

4.2.2 Presenter Example

- I have students run a brief, practical statistical analysis using SPSS (proprietary) *or* R (FOSS).
- For both options I:
  - Provide a video and written guide on how to perform the essential skills for the assignment
  - Demonstrate equivalency of outcomes between the two programs, e.g., I show a t-test result is the same between both of the programs
  - It's important that you make it clear that both pieces of software offer a reasonable way to accomplish the same task
- For either option, students must:
  - Show output sceenshot from the program of their choice
  - Produce the same outcome in quality of work, regardless of program choice
- Depending on complexity of assignment, I may offer a small amount of extra points for use of R, due to it generally requiring a higher learning curve.

4.2.3 Benefits

- Gives agency for students to choose which path they would like to take, allows for students to pursue the option they feel is more valuable / attainable
- Ideally, students demonstrate same skills and accomplish same learning outcomes, regardless of chosen option
- Demonstrates instructor ability / competence, show your own ability to accomplish the work, model problem solving behavior across multiple modalities

4.2.4 Drawbacks

- Additional work for instructor to prepare guidance and instructions for two different programs
- Students may chose the perceived "easier" option - that is why it is important to try and find equivalent options
- Students may switch their software choice halfway through, if they feel they made the "wrong" choice, doubling the amount of time the assignment takes and causing frustration

4.3 Option 2: A Shorter Assignment, Both Ways

4.3.1 Steps

1. Think of an assignment you already give to your class in which you recommend/suggest/require a proprietary software
2. Rewrite *and shorten* assignment instruction to allow for use of the proprietary *and* the open-source software alternative. *Very important to emphasize what purpose doing the same assignment twice serves.*
3. Have students progress through the same assignment twice, demonstrating the same skills across both software platforms, and ensuring they get equivalent results across the two

4.3.2 Presenter Example

- Students need to record a short introduction (to themselves) video in a class in which they will have to do multiple recording presentations. They are asked to record once using Panopto (Proprietary) and Open Broadcasting Software (OBS; FOSS)
- For both options I:
  - Provide a video that shows navigating and recording a video with the selected program
  - Show how both produce the same looking and sounding video, but require different steps to configure and set up
- For both options, students must:
  - Record an equivalent video, producing the same outcome in quality of work, regardless of program choice

4.3.3 Benefits

- Strongly reinforces the idea of different method/software, same outcome
- Helps students appreciate learning objectives as being accomplish-able through multiple ways, avoiding a “one-track” mindset

4.3.4 Drawbacks

- May easily be grating and boring to students if assignment is too long and feels like too much work to do twice
- Students may not see value in multiple method approach unless this is explicitly explained when introducing assignment

4.4 Option 3: Scavenger Hunt for Equivalent (or Different) Features

4.4.1 Steps

1. This works best if used later in the course after students have already established reasonable competence with a proprietary program
2. Think of a set of *simple* practices / skills that exist as part of the proprietary software program that your students *already know how to do.* E.g., how to crop a photo, how to find a mean, etc.
3. Encourage students to explore a similar, equivalent open-source program and determine how to accomplish those same tasks - encourage use of Google, ChatGPT (within reason), and documentation to find. Have them take screenshots showing them "finding" the features in the open source program.
4. (Optional) Ask for evidence of a simple demonstration of those found features
5. (Optional) Have them find a feature present in one software, but NOT the other

4.4.2 Presenter Example

- In a formula-heavy class, students are primarily use to hand-writing or using formula functions built-into MS Word or Google Docs, but I would like them to practice writing formulas in LaTeX
- I request that students find the correct LaTeXnotation for writing common formula notation like superscripts, subscripts, fractions, summations, etc.
- I have them send me links to the resources they used and write out some simple examples of formulas in LaTeX, using the advice from those resources

4.4.3 Benefits

- Encourages autonomy in students to find and use resources, even when not directly provided by instructor
- Stresses problem-solving, savvy searching, and application of solutions
- Uses their established competence in one program to help connect to a new program
- With optional step 5, may help students understand relative limitations of certain software choices

4.4.4 Drawbacks

- Students *must* feel competent in completing these practices already, otherwise may feel like they are trying to learn tech and concepts at the same time, which may be

overwhelming
- Depending on student background, the instructor may have to provide some hints as to where to look to find solutions - some students may find the “scavenging” part to be unsupported by the instructor (even if that is intentional!)

> **Important**
>
> There are several routes to integration of open-source tools, but all focus on flexibility and agency!

5 Conclusion

5.1 Recap

- There are many reasons we should be conscious around what software we use, demonstrate, and require in instruction, especially with the growing cost and monopoly certain tools
- FOSS software *can*, and sometimes *should*, be considered as an alternative when training students for practical work - it may also offer an opportunity to practice soft problem-solving and adaptation skills vital for the workplace
- *Exposure and experience is key*: we don’t necessarily need to persuade students of a certain method of using software, so much as we should help them explore their options, for both now, and in the future

5.2 Parting Message

- Continue using the resources and links provided throughout this presentation to explore the complexity around adopting more open-source software
- If you are an instructor: consider trialing an adoption of open-source tools, like one of my Examples of Integration, in your classrooms or, maybe even in your own workflow
- If you are administrative or support staff, make sure that instructors are aware of alternative options to the most popular tools.
- There is more to consider that we didn’t even cover today, like possible transparency, reproducability, and security benefits that may come with adoption of open-source software. Consider looking into these topics further!

- Encourage students to engage fruitfully with software not purely as a means-to-an-end, but as a **conscious and adaptive choice** - share with them some ideas present in this presentation!

5.3 Follow-Up

- Thanks for joining me today!
- If you want to stay in-touch and talk more, send me an email at QuagliaQ@gvsu.edu, or catch me after the presentation!

5.4 References

Allaire, J., & Dervieux, C. (2025). *Quarto: R interface to quarto markdown publishing system*. https://github.com/quarto-dev/quarto-r

R Core Team. (2025). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing. https://www.R-project.org/

Xie, Y. (2014). Knitr: A comprehensive tool for reproducible research in R. In V. Stodden, F. Leisch, & R. D. Peng (Eds.), *Implementing reproducible computational research*. Chapman; Hall/CRC.

Xie, Y. (2015). *Dynamic documents with R and knitr* (2nd ed.). Chapman; Hall/CRC. https://yihui.org/knitr/

Xie, Y. (2025). *Knitr: A general-purpose package for dynamic report generation in r*. https://yihui.org/knitr/