QUARTO THE ASSASSIN

JEFFREY S. RACINE

Abstract. My interest in open source research tools has a long arc, and the tools I adopt evolve as new and improved approaches surface (Racine and Hyndman 2002; Meredith and Racine 2009; Racine 2012). A recent development brings to mind overused superlatives like "game-changer", among others, and I suspect you will agree "100%" that one recent development belongs in either the "Assassin" or "Saturn" category, or perhaps both (the latter being associated with consuming one's progeny as prophecy had it that Saturn would be overthrown by one of his sons, so in response, he devoured them upon birth). In this Chapter we examine how Quarto can be used to create dynamic reveal.js presentations that are simply unrivalled, anywhere, period ("100%", a "game-changer"). As such, and particularly for those accustomed to the Beamer slide format, the adoption of these new tools may be of particular interest. This document, naturally, is authored in Quarto, and the Quarto script for this document and the reveal.js slides discussed below are available at https://github.com/JeffreyRacine/rch.

1. TEX, Beamer, and the Pedagogy of Econometrics

This Festschrift, a compilation intended to honor R. Carter Hill whose lifelong passion for the discipline has inspired generations, centers around teaching econometrics and the pedagogics of econometrics. Carter Hill remains a highly influential and prolific scholar, and a small sampling of his noteworthy contributions would include Griffiths and Hill (2022b), Griffiths and Hill (2022a), Martin and Hill (2022), Zeng and Hill (2020), Campbell and Hill (2006), Hill, Knight, and Sirmans (1997), Deis and Hill (1995), Kim and Hill (1995), Hill (1994), Hill, Cartwright, and Arbaugh (1991), Hill and Judge (1990), Judge, Hill, and Bock (1990), Griffiths, Hill, and Pope (1987), Hill and Judge (1987), Hill and Ziemer (1984), and Fomby, Hill, and Johnson (1978), to name but a few of the peer reviewed publications he has authored to date.

The pedagogics of econometrics, at its core, involves the transmission of mathematics-based knowledge. Econometricians often use slide presentations to transmit knowledge to their audience, Microsoft PowerPoint and LaTeX Beamer being perhaps the two most popular tools in use today. Though PowerPoint is an industry standard, it is a closed-source commercial application and some econometricians recoil at using it for presenting academic research, and for good reason - PowerPoint is not a typesetting program, and the mathematics it produces appear primitive to many, myself included. And this is why open source Beamer is so appealing to

Date: 2024-08-18.

some as it is member of the TEX typesetting system (strictly speaking, Beamer is part of LATEX which is a superset of TEX). TEX is an open source de-facto global standard for typesetting mathematics that has been universally adopted by econometricians and many others (it was written by Donald Knuth and initially released in 1978 (Knuth 1978)).

Though PowerPoint and Beamer have been around for decades, a more recent arrival to the presentation arena called 'reveal.js' is generating a fair bit of justifiable buzz. While Beamer generates PDF output with perfectly typeset mathematics generated by TEX, reveal.js generates Hyper Text Markup Language (HTML) output which does not guarantee that non-ugly mathematics will be generated. But reveal.js can exploit MathJax, and thanks to MathJax's support of TEX you can effortlessly obtain typeset mathematics in the HTML output that is *identical* to what you would get using Beamer (you view HTML output in a web browser, and MathJax is a display engine for mathematics that works in all browsers).

Authoring slides natively in Beamer and reveal.js can be quite tedious (the former requires command of the TEX programming language, the latter the JavaScript programming language). So, when R Markdown arrived on the scene in 2014 and allowed one to author Beamer slides using this lightweight scripting language, it was widely embraced. Quarto Markdown is a new kid on the block, relatively speaking, and was publicly released in 2022. Quarto Markdown is a lightweight language virtually indistinguishable from R Markdown that can be used to author reveal.js slides (and Beamer slides, etc.), and it turns out that Quarto Markdown is the progeny of the same group of developers that birthed R Markdown. Are there any benefits to migrating to reveal.js from Beamer, or is it just another interchangeable tool one could adopt to accomplish the same end result? This Chapter attempts to definitively answer this question.

2. Quarto - A Familiar Framework for Many

Quarto is a framework for reproducible document creation in which you weave together both your narrative and program code into a document which can be rendered into a broad array of output formats (see quarto.org for additional details). If this sounds familiar, it is because Quarto Markdown is an evolution of R Markdown. Quarto was only recently released, has evolved rapidly, and is now quite stable (version 1.0 was released publicly July 28 2022 - see the Quarto Announcement for details).

Quarto corrals the various enhancements to R Markdown that reside in different libraries, packages, and extensions, and brings them together under one roof. What sorts of enhancements are we talking about? Things like cross-referencing that are now widely portable across output formats, for one (missing in R Markdown, requiring R Bookdown extensions). Furthermore,

we are no longer restricted to using the R language or the RStudio integrated development environment (IDE) as there is a stand-alone Quarto command line interface (CLI) available for all platforms, while support for a range of features now exists that previously did not. But under the hood the basic architecture remains largely intact, with reliance on Pandoc being indispensable for rendering Quarto scripts into usable output in a variety of formats. As well, support for typesetting mathematics using TEX and for bibliography management using BibTEX remains intact.

Quarto Markdown, which is almost identical to R Markdown, is a simple plain text format designed to be easy to write and, more importantly, easy to read. A Markdown-formatted document should be publishable as-is, as plain text, without looking like it's been marked up with tags or formatting instructions. That is, markup languages such as HTML are known to be *heavy* while markdown languages such as Quarto Markdown are know to be *light*. So light, in fact, that they embody some of the simplest ways of creating documents one could possibly imagine.

But the "hidden superpower" of Quarto Markdown, as is the case for R Markdown, is its support for output formats generated on-the-fly by Pandoc (https://pandoc.org). Pandoc is an almost magical tool for converting documents from one format to another. You author your document using Quarto's markdown language (simple text), then *render* it in one of a dizzying number of potential document formats, and Pandoc does its conversion magic *in the background*. It truly could not get any easier thanks to Quarto's developers who have put an incredible amount of time and energy into creating this next-generation open source scientific and technical publishing system.

This Chapter is not going to focus on Quarto, however. Quarto will be a very familiar tool to those of you at all familiar with using RStudio and R Markdown to generate reproducible documents, write technical articles, or create presentations. This Chapter will focus instead on something that has flown under the radar but which I only recently decided to test drive, and even though it is under active development and currently has a few areas in need of further development, it is definitely ready for prime time (by the time this Chapter appears it almost certainly will be fully mature). And you do not want to stand around any longer using R Markdown/Beamer or other such tools for creating reproducible slides, unless you harbor masochistic tendencies. The benefits of switching to reveal.js are far too many to ignore, while the uninitiated reader will appreciate that there is a new kid in town who deserves all the attention they receive.

3. reveal.js - A Bridge Too Far?

A few words on reveal.js are in order. The reveal.js presentation framework (https://revealjs.com) is an open source platform for creating presentations in HTML. reveal.js enables anyone with a web browser to create dynamic feature-rich presentations for free. Presentations made with reveal.js are built on open web technologies, so *anything* you can do on the web, you can do in your presentation. These dynamic presentations allow for a range of features that simply cannot be embedded in static presentation platforms such as PowerPoint or Beamer.

However, in the not-too-distant past, generating reveal.js presentations required knowledge of HTML and the computing chops necessary to install and build software locally. Though the reveal.js website documentation is very well structured and written, both the installation of the resources *and* authoring in HTML put reveal.js beyond many users' reach. HTML is the standard *markup* language for Web pages, and is fairly heavy with lots of syntax that one must master in order to be proficient.

But sorrow can turn to joy, as is the case with Quarto Markdown and its support for reveal.js which operates in a remarkably similar way to R Markdown and Beamer (you would be hard pressed to distinguish the R Markdown/Beamer script from the Quarto Markdown/reveal.js script at a distance). Both use a standard YAML header (metadata), simple hashtags to separate slides, dashes to delineate bulletpoints on each slide, support for TEX and BibTEX, etc. But unlike Beamer formatted output, which produces a *static* PDF document, reveal.js formatted output produces a *dynamic* document in HTML format. And producing either format is as simple as opening a new file, adding some plain text, then clicking on a button in RStudio (I recommend starting with RStudio as this environment is totally self-contained and ready-to-go).

Unlike its static Beamer peer, the dynamic document generated by Quarto and reveal.js is, well, dynamic! That's because this process involves JavaScript, which is a computer programming language commonly used to create interactive effects within web browsers. But the beauty of this framework is that you don't have to know one iota about programming in HTML nor coding with JavaScript. Quarto Markdown takes care of all of the complex details!

The bottom line is that if you are familiar with R Markdown, you will be in your element using Quarto Markdown. And if you have used R Markdown to generate Beamer slides, you will be totally at home using Quarto Markdown to generate reveal.js slides. And if you already use RStudio to author R Markdown documents including Beamer output formats, you can use

RStudio to author Quarto Markdown documents that now include reveal.js among your output format options.

So what is all the fuss about? After all, Beamer is a brilliant tool for creating presentations that, when combined with R Markdown, allows you to weave your code and slide narrative into one document leading to a streamlined workflow. You only work with one document (the R Markdown script), you write using simple Markdown (trivial to master), and you can strip off your R code for distribution on its own with a simple command (i.e., using tangle()). But, at the end of the day, what you get is a PDF document which is a *static* output format (rendered via Pandoc that converts the Markdown script into Beamer slide format that automatically is compiled as a PDF).

By switching from R Markdown/Beamer to Quarto Markdown/reveal.js you can create, by way of illustration, live tabs with hidden tables that you can click on to reveal content dynamically. Or, you can embed shiny code directly in the document so that the presenter and audience can conduct analysis on-the-fly. Another impressive aspect is under-the-hood support for *multiplexing*, which is a wickedly good and trivial to use framework whereby your audience can be viewing slides on their mobile device (phone, tablet, laptop etc.) that responds to your, the presenter, gestures (incremental advance of each slide, say) in *real time* with *zero* setup required by your audience (you simply start your talk with a link to the slides that can be hosted for free by, say, GitHub Pages, the audience follows that link to view the slides locally and that's it). Any comments made by audience members can be incorporated into the slides almost immediately should you choose to push said changes to GitHub Pages (where they will typically be deployed in under a minute or so), while they can submit an "Issue" on the GitHub repository for you to assess, or they could even "push" a suggested change that you could "merge" after consideration. All in real-time.

So, let's dispense with the modern history lesson and hype and dig right in. In what follows I hope to convince you that if you already use R Markdown/Beamer then the cost of migrating to Quarto Markdown/reveal.js is truly trivial, while the benefits are simply too great to be discounted or ignored.

4. Getting Started

This section will walk you through getting a working installation of Quarto on your computer and then using a pre-existing sample reveal.js "toy example" to test your system and set the stage for unlocking the potential of reveal.js presentations.

4.1. **A Minimal Installation.** Perhaps the easiest way to get started is to install R, RStudio, and TEX. Whether you are using Microsoft's Windows, Apple's macOS, or Linux/UNIX, the cross-platform open source and freely available components outlined below ought to get you headed in the right direction.

> **!** Important
>
> - Quarto was recently released, has evolved rapidly, and is now quite stable (Version 1.0 was released publicly July 28 2022 - see https://posit.co/blog/announcing-quarto-a-new-scientific-and-technical-publishing-system)
> - However, *older versions of RStudio do not support Quarto* (only RStudio v2022.07.1 and later includes support for editing and preview of Quarto documents)
> - Therefore, it is *strongly* advised that, even if you have versions of R and RStudio already installed on your system, you *install the most recent versions*

1. R and RStudio (see https://posit.co/download/rstudio-desktop)

   > **!** Important
   >
   > - You *must* install R before installing RStudio (otherwise RStudio will not function, so the order of installation is critical)
   > - If you are using macOS (i.e., an Apple laptop), when you run the RStudio installer you need to *manually* copy/drag the RStudio app to the Applications folder

2. TEX

   > **!** Important
   >
   > In order to create PDFs and typeset mathematics you will need to install a recent distribution of TEX. Quarto developers recommend the use of TinyTEX (which is based on TEXLive), which you can install from the *Terminal in RStudio* as follows; open RStudio, select the menu *Tools -> Terminal -> Move Focus to Terminal*, then type the following command at the *Terminal* Prompt (i.e., inside the *Terminal pane* in RStudio):
   >
   > *quarto install tinytex*

If you instead wish to install a standalone version of TEX, Windows users can see miktex.org, macOS users can see https://www.tug.org/mactex, and Linux/UNIX users can see https://www.tug.org/texlive.
Note - you must install one or the other, so if you already have TEX installed you should be good to go (they may not coexist harmoniously on your system).

4.2. **A Simple reveal.js Presentation - Testing Your Installation.** Once you have installed R, RStudio, and (optionally for some) TEX, you should be ready to create your first deck of reveal.js slides. Let's get going.

- Open RStudio (Figure 1).

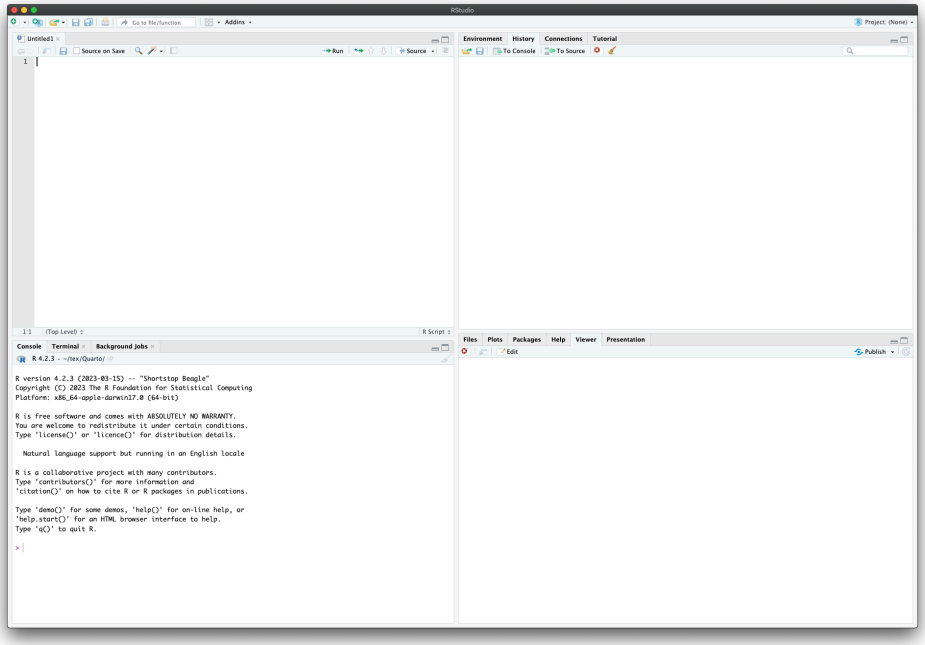

FIGURE 1. RStudio

- From the File Menu in RStudio, select File -> New File -> Quarto Presentation. This should present you with a dialog with reveal.js highlighted, then click the Create button (Figure 2). Name your file, then click the Render icon located in the upper portion of the RStudio editor (with the blue arrow immediately to its left). You may be presented with a request to install certain packages, which you must do and can accomplish by simply selecting the install icon that is presented.

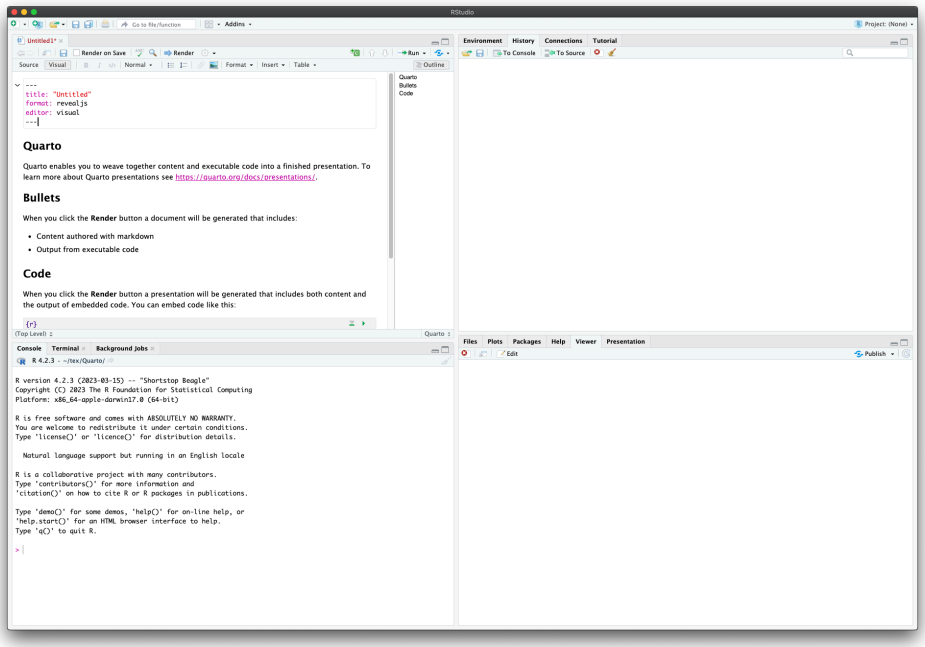

FIGURE 2. RStudio Sample reveal.js Presentation

- By "rendering" your document, a presentation will open up in the presentation window below and to the right of your editor window, by default (Figure 3). Using your arrow keys with your mouse focus on the viewing window, you can navigate to subsequent slides in the sample presentation (note that the viewing window, visible in Figure 3, is the pane on the lower right with the word **Untitled**).

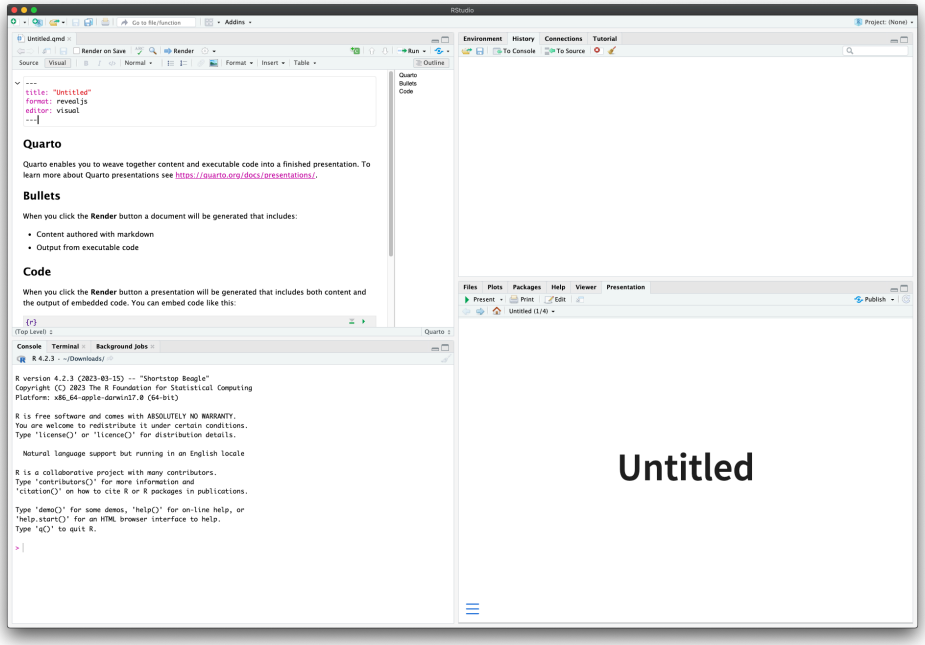

FIGURE 3. RStudio Sample reveal.js Presentation Rendered in Viewing Window

If you have gotten this far, sit back and congratulate yourself! You are well on your way to mastering an exciting new dynamic way of presenting information to your intended audience that is capable of going far beyond what you achieved when using static presentation platforms such as LaTeX Beamer.

Now that you have installed a functioning system for creating documents using Quarto, let's start once again from scratch and delve into the power of reveal.js presentations.

4.3. **Becoming More Familiar with reveal.js Presentations.** As you can see in Figure 1, the first five lines (the dashes are treated as a line) in the default Quarto document are

```
---
title: "Untitled"
format: revealjs
editor: visual
---
```

This chunk of code, which begins any Quarto Markdown document, is the "YAML Header", (YAML stands for "yet another markup language"). This can be thought of as the "metadata" that defines the document, and can be modified and extended to accomplish virtually anything

you wish. If you modify the editor line and change `visual` to `source` this will reveal the Quarto Markdown source code (i.e., switch the editor from visual to source mode, which can also be accomplished by toggling the source and visual buttons in the upper left of the editor screen by default). Either way, viewing the Quarto Markdown file in source mode ought to produce Figure 4.

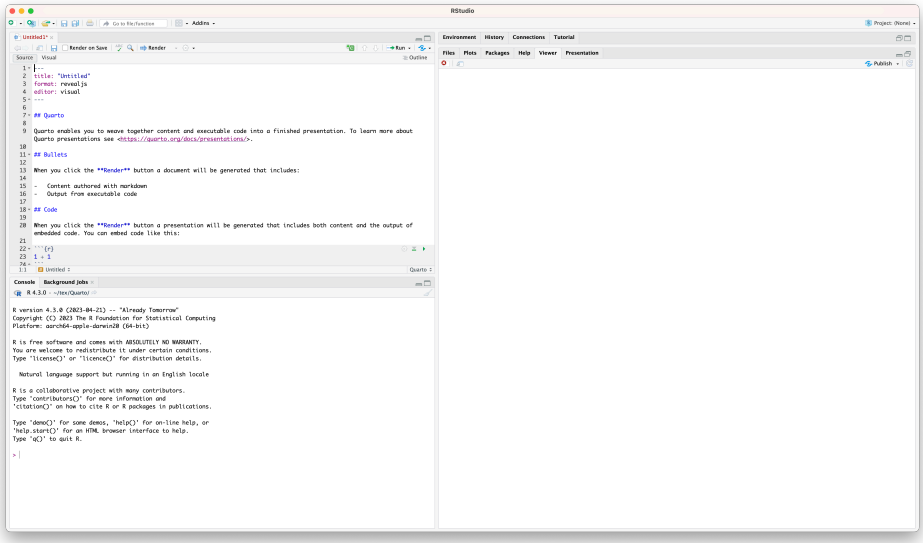

FIGURE 4. RStudio

Figure 4 reveals the full "raw" contents of this simple example of a Quarto Markdown document. First there is the YAML header indicating the title, format of the output desired, and (optionally) whether you wish to use the source or visual editor interface while editing. These YAML instructions are separated from the document narrative, which appears beginning on Line 6 and onward (Line 6 is a blank line - white space and line breaks are extremely useful in lightweight languages such as Markdown). Line 7 is a section header, and hashtags that begin lines indicates sections and subsections if you were authoring, say, an article or book (two hashtags in a slide presentation indicate the slide title, while one hashtag would indicate a section which is displayed on its own slide). Lines 7-10 contain information that will appear on the second slide (the first will be the title slide), lines 11-17 information that will appear on the third slide, etc. Note that lines 15 and 16 start with a dash, which indicates an unnumbered list. If you started lines 15 and 16 instead with , say, 1. they would appear as a numbered list, i.e., if you modified lines 15 and 16 to read instead

```
1. Content authored with markdown
1. Output from executable code
```

Note that here the use of `1.` simply indicates the presence of a numbered list, it will be properly numbered in ascending order in your output document (that is, this is a simple instruction to Quarto Markdown indicating each element is part of a group of numbered lists).

Now, for those of you not familiar with R Markdown and Quarto Markdown documents, lines 22-24 contain the "secret sauce" underlying reproducibility. This chunk contains some instructions in the R Language (though if instead Line 21 contained the string `python` instead of `r` it would be interpreted instead as some instructions in the Python language). The "secret" here is that when you render your Quarto document the code in such chunks will be *automatically* executed and the resulting output inserted into your document, with or without the code producing the output appearing in the slide, as desired). So, if your code was `hist(rnorm(100))` instead of `1+1`, a figure would be produced instead of the addition of the two numbers. And if you don't want the code itself to be echoed in your output document but only the results, you can add `#| echo: FALSE` as the first line of code in the code chunk and the R code will not appear, instead only the output generated by the code. In short, this is why so many researchers have gravitated towards using Quarto and R Markdown for their research - you only work with one file that contains both your narrative and code, and can be agnostic about the output format right up until you choose to generate your preferred output format. And your figures (and tables etc.) are generated on the fly and inserted directly into your document without any copying and pasting. Then there is the support of TEX for mathematics, cross-referencing, and so much more.

Once you arrive at this point, you might be sold on the virtues of using Quarto or R Markdown to generate your Beamer output format slides, but may be wondering why instead you might use the reveal.js output format? We now examine the additional flexibility afforded by doing so that simply cannot be replicated if you restricted yourself to using Beamer format PDF slides. Furthermore, you can see how effortless it is to move between the two formats by simply changing `revealjs` on Line 3 to `beamer` and re-rendering the output.

5. revealjs - The Markdown Beamer Assassin

When you adopt the revealjs format in Quarto Markdown documents, Quarto will generate an HTML document which you can subsequently view with any browser. So, if you wanted to present your slides to an audience you simply open your HTML document with a browser and

begin (with, say, Beamer, you would have a PDF document which you subsequently view with a PDF viewer such as Adobe Acrobat). Here the similarities end, since PDF is static while HTML is dynamic. We now enumerate a few of the benefits of using reveal.js over Beamer PDF slides.

5.1. **Speaker Notes.** One benefit missing from Beamer PDF slides is the ability to add speaker notes. To do so you simply add the following code to your markdown slides:

```
::: {.notes}
Speaker notes go here.
:::
```

Then, when viewing your presentation in a browser, you press the `S` key (or use the Navigation Menu) to display the presentation speaker view on your monitor.

5.2. **Copy to Clipboard.** If you have code chunks in your document and want the audience to be able to easily access the code, you can use the `copy to clipboard` button for code blocks (upper right in block) to copy and paste into the desired location. This is automatically applied to all visible code chunks. Of course, this presumes your audience has access to your slides, which is trivial to accomplish using GitHub Pages by way of illustration (see below where we discuss hosting your slides on GitHub).

5.3. **Zoom.** You can zoom in and out (i.e., magnify) in by holding down the Alt key in MS Windows, Opt key in macOS or Ctrl key in Linux, and clicking on any screen element (Alt/Opt/Ctrl click again to zoom out).

5.4. **Annotation and Chalkboard.** You can pull up a chalk board by pressing the B key (B to revert), the annotation (notes) canvas by pressing the C key (C to revert), can press the Del key to erase, and can press the D key to download drawings (stored in json format).

5.5. **PDF Export.** Should you (or your audience) wish to save a PDF copy of your slides, you press the `E` key (wait a few seconds, then print [or print using system dialog], enable landscape layout, then save as PDF - press the `E` key to revert).

5.6. **Multiplexing.** Multiplexing is one of the more compelling reasons to consider adopting the reveal.js output format in my opinion. When enabled, rendering your slides will create two documents, `foo.html` and `foo-master.html`. Anyone who has a copy of `foo.html` and is connected to the internet will find that their slides are `multiplexed` with the slides `foo-master.html`. This means that, almost magically, as the presenter advances the slides in `foo-master.html`, the participant who has `foo.html` open on their computer in a browser will

have their presentation *automatically* advance as the speaker proceeds through their presentation. So, anyone anywhere is linked to the master presentation in real time, whether they are in the back of an auditorium and unable to see details on the projector screen in the front of the auditorium or prefer their laptop screen to the projector view. Enabling this is done by adding an option to the revealjs format in your YAML header as follows:

```
format:
  revealjs:
    multiplex: true
```

The amount of internet plumbing activated by this simple command yet hidden from the user is truly stunning, to my way of thinking.

5.7. **Hosting Slides via GitHub.** Once you have created your slides, there is the issue of how best to access them, particularly with audience participants you wish to engage via multiplexing.

One solution I find particularly useful is to host the slides on GitHub (https://github.com) via a free service called GitHub Pages (GitHub Pages serves up a website, and your HTML presentation is effectively a website; see https://pages.github.com for details). If you title your reveal.js document `index.qmd`, then by default, a document titled `index.html` is located in a GitHub repository that you create, then you enable its hosting by activating this HTML document (in your GitHub repository click the `Settings` icon, then `Pages` then `Deploy from Branch` and select `Main`, and after a minute or so you will see a message `Your site is live at https://yourname.github.io/foo/` where `foo` is the name of the repository. Note you will upload to GitHub your `index.html` *plus* the subdirectory `index_files` which contains the auxiliary files generated including various JavaScript objects required to properly render your site. Then, you can begin your presentation with a slide containing the name of your GitHub Pages repository slides, and when you run the master slides `index-master.html` on your computer, multiplexing should be automatically enabled.

I must confess that students and colleagues seem to genuinely love this feature and, for teaching econometrics, having code chunks available via cut-and-paste along with multiplexing (which encourages students to stay on task) in my experience has been extremely positive. There is nothing I have been able to do in the past that is as streamlined as this solution and it is difficult to envision it being made any simpler. In addition, RStudio supports GitHub natively, so you can push your changes to GitHub directly from RStudio with a few clicks (to enable this you simply create a project in RStudio then enable GitHub for your project, then you can synchronize your document with GitHub with the click of a button).

Some unsolicited advice here may be in order. Don't be put off if you are unfamiliar with GitHub. There are excellent introductory how-to videos available, but the best resource will be a student or colleague already familiar who can spare two minutes and get you started. The learning curve is not steep, and the benefits are substantial. And, of course, you don't need to use GitHub to experience the full benefits of reveal.js, but it is a very nice feature to have available.

6. Taking reveal.js for a Test Drive

OK, enough with the sales pitch, let's get our hands dirty. The following YAML header can be used to generate a reveal.js presentation with a few additional bells and whistles enabled, including multiplexing and other useful options mentioned above. You can download the full Quarto Markdown reveal.js script from https://github.com/JeffreyRacine/rch then render the document in RStudio to see the results for yourself (see the README section for further details). If you successfully render the document, you will have a working reveal.js presentation with title slide similar to that in Figure 5, and you can then begin to explore the various features available to you. The document contains a number of examples that you can modify to test various features, and you can walk through the document to see how the various features are implemented, most importantly, those relying on dynamic content not available in, say, Beamer PDF slides.

```
---
title: Festschrift? Gesundheit!
author: R. Carter Hill
institute: Louisiana State University
date: today
editor: source
bibliography: rch.bib
format:
  revealjs:
    background-transition: fade
    css: rch.css
    center: true
    chalkboard: true
    footer: Arrr, Get Yer Own Damn Festschrift | R Carter Hill
    html-math-method: mathjax
    include-in-header:
      - text: |
          <script>
          window.MathJax = {
            tex: {
```

```
            tags: 'ams'
          }
        };
        </script>
    incremental: true
    multiplex: true
    slide-number: true
    theme: default
    transition: slide
---

Additional content in the online document is not displayed here for brevity.
```

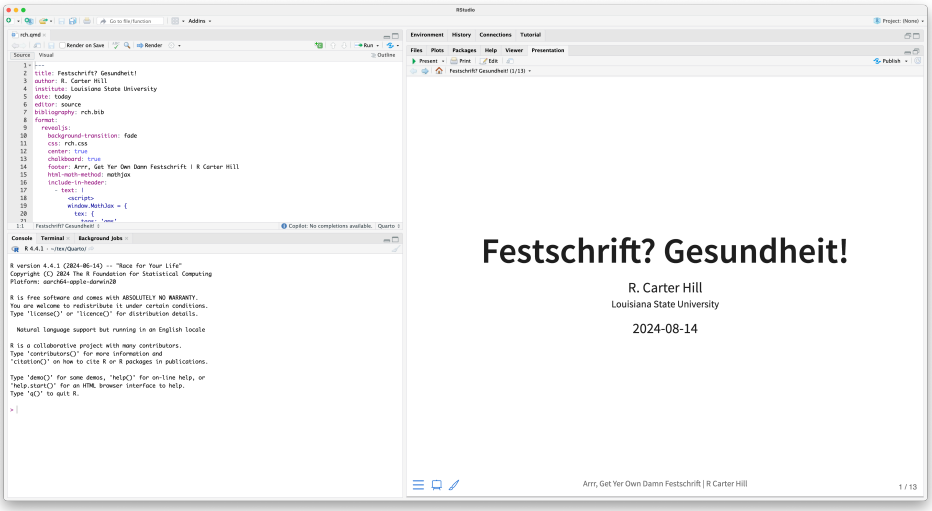

FIGURE 5. Festschrift Slides

6.1. **A Few Points To Note.**

- You will notice a board and marker icon in the lower left of the document when you preview it (in RStudio or a browser) which you can click on to enable. Needless to say, this is a very powerful feature that is trivial to enable, and is a feature that is not available natively in Beamer PDF slides.
- You might have noticed the option `touch: true`, which enables touch navigation on devices with touch input (e.g., smartphones and tablets). This is a very nice feature to have available, and is not available in Beamer PDF slides.
- You will also see that two files were created (since multiplexing was enabled), `foo.html` and `foo-master.html` (where `foo` is the name of your document, in this case `rch`). You

can open `foo.html` in a browser and `foo-master.html` in another browser window, and as you advance the slides in `foo-master.html` the slides in `foo.html` will advance in real time. This is a very powerful feature that is trivial to enable, and is a feature that is not available in Beamer PDF slides.

If you have made it thus far and have successfully rendered the document, you are now in a position to begin to explore the various features available to you in reveal.js presentations. A great resource can be found at https://quarto.org/docs/presentations/revealjs/ which will provide you with a comprehensive overview of the features available to you in reveal.js presentations.

7. Summary

I am grateful to R Carter Hill for his inspiration and guidance over the years, and I hope that this Chapter will serve as a fitting tribute to his many contributions to the field of econometrics. This Chapter has been my attempt to introduce you to Quarto Markdown and reveal.js, two tools that are revolutionizing the way we create and present documents (you can download the Quarto Markdown files for this Chapter and the reveal.js slides from https://github.com/JeffreyRacine/rch). Quarto Markdown is a lightweight language that is easy to write and read, and is capable of generating a wide range of output formats, and reveal.js is a presentation framework that allows you to create dynamic feature-rich presentations in HTML. By combining Quarto Markdown and reveal.js, you can create presentations that are simply unrivalled, anywhere, period. If you are already using R Markdown/Beamer to create your presentations, the cost of migrating to Quarto Markdown/reveal.js is trivial, while the benefits are simply too great to be discounted or ignored. I hope that this Chapter has convinced you to give Quarto Markdown and reveal.js a try, and that you will find them as useful and powerful as I have.

References

Campbell, R. C., and R. C. Hill. 2006. "Imposing Parameter Inequality Restrictions Using the Principle of Maximum Entropy." *Journal of Statistical Computation and Simulation* 76 (11): 985–1000.

Deis, D., and R. C. Hill. 1995. "An Application of the Bootstrap Method to the Simultaneous Equations Model of the Demand and Supply of Audit Services." *Contemporary Accounting Research* 15 (1): 83–99.

Fomby, T. B., R. C. Hill, and Stanley. R. Johnson. 1978. "An Optimality Property of Principal Components Regression." *Journal of the American Statistical Association* 33: 191–93.

Griffiths, W. E., and R. Carter Hill. 2022a. "On the Power of the F-Test for Hypotheses in a Linear Model." *The American Statistician* 76 (1): 78–84. https://doi.org/10.1080/00031305.2021.1979652.

———. 2022b. "Rejoinder to Harville (2022) and Christensen (2022) Comments on 'on the Power of the F-Test for Hypotheses in a Linear Model,' by Griffiths and Hill (2022)." *The American Statistician* 76 (3): 312.

Griffiths, W. E., R. C. Hill, and P. Pope. 1987. "Small Sample Properties of Probit Model Estimators." *Journal of the American Statistical Association* 82: 929–37.

Hill, R. C. 1994. "The Statistical Properties of the Equity Estimator." *Journal of Business and Economic Statistics* 12: 141–49.

Hill, R. C., P. Cartwright, and J. Arbaugh. 1991. "The Use of Biased Predictors in Marketing Research." *International Journal of Forecasting* 7: 271–82.

Hill, R. C., and G. G. Judge. 1987. "Improved Prediction in the Presence of Multicollinearity." *Journal of Econometrics* 35: 83–100.

———. 1990. "Improved Estimation Under Collinearity and Squared Error Loss." *Journal of Multivariate Analysis* 32: 296–312.

Hill, R. C., J. Knight, and C. F. Sirmans. 1997. "Estimation of Capital Asset Price Indexes." *Review of Economics and Statistics*, May, 226–33.

Hill, R. C., and R. Ziemer. 1984. "The Risk of Stein-Like Estimators in the Presence of Multicollinearity." *Journal of Econometrics* 25: 205–16.

Judge, G. G., R. C. Hill, and M. E. Bock. 1990. "An Adaptive Empirical Bayes Estimator of the Multivariate Normal Mean Under Quadratic Loss." *Journal of Econometrics* 44: 189–213.

Kim, M., and R. C. Hill. 1995. "Shrinkage Estimation in Nonlinear Regression." *Journal of Econometrics* 66: 1–35.

Knuth, Donald E. 1978. "TeX." Computer program.

Martin, R. E., and R. C. Hill. 2022. "Baumol and Bowen Cost Effects in Research Universities." *Southern University College of Business E-Journal* 17 (1).

Meredith, E., and Jeffrey S. Racine. 2009. "Towards Reproducible Econometric Research: The Sweave Framework." *Journal of Applied Econometrics* 24: 366–74.

Racine, Jeffrey S. 2012. "RStudio: A Platform-Independent IDE for R and Sweave." *Journal of Applied Econometrics* 27 (1): 167–72. http://www.jstor.org/stable/41337225.

Racine, Jeffrey S., and R. Hyndman. 2002. "Using R to Teach Econometrics." *Journal of Applied Econometrics* 17 (2): 175–89.

Zeng, T., and R. C. Hill. 2020. "Stein-Rule Estimation in Genetic Carrier Testing." *International Journal of Computational Economics and Econometrics* 10 (2): 111–28.