Plugins and Filters and Data Science: Displaying R Markdown in Drupal

Today we are going to talk about how we render dynamic data documents in Drupal 8. We are going to talk about this because R Markdown (with the help of knitr and pandoc) is very good at generating "publish-ready" documents, well-formed HTML, and even fully-functional, multi-page Web sites. We are going to talk about this because, at Urban, most research documents are NOT published as standalone Websites. We are going to talk about this from the perspective of the publisher.

We are going to talk a little about Drupal, a little about markdown, and a little about PHP. These things may or may not be of interest to you as a data science blog reader. That's ok. The Urban team has plenty of data science red meat planned for this space. On the off-chance you are interested in seeing how Drupal and R markdown might work together, you have come to the right place.

Urban is a Drupal shop.

There's no getting around the need for some sort of "Why we chose Drupal" component of this post. It should be easy, but it wouldn't be honest. The truth is we chose Drupal as our primary Content Management System (CMS) several years ago. The reasons for this decision could fill another blog post (and perhaps one day they will).

Our Web Development team currently "owns" 15 (and counting) production Drupal installations. We've invested in the platform by keeping Drupal expertise in-house. This allows us to stay nimble and roll-out new tools and features throughout the year. It also provides a growing knowledge base that keeps us on the forefront of digital research publishing.

So, as requirements are gathered, if there is a CMS component, we will default to the Drupal platform unless there is a compelling reason to go in a different direction (you'll notice we aren't forcing cloud-based Spark microservices through a Drupal stack). We determined NCCS would start with a CMS (and, therefore, Drupal) due to the following requirements:

- NCCS staff to manage all site content, including to add new publications as needed.
- NCCS staff can add users and assign permissions to add/edit site content.
- Content can be tagged and categorized.
- The site includes a keyword search and publications must be searchable.
- The site, and all publications therein, must adhere to Urban brand standards.
- The site must support markdown.
- Since code will be displayed as content, the site must provide standard syntax highlighting on both the backend (where content is created) and the front-end (where it is displayed).
- Publications must support a fixed-column Table of Contents (TOC), automatically generated from the header tags within the publication text.
- R markdown often generates graphic assets (images, charts, etc). It must be easy for a site editor to upload related assets without having to re-create the publication within Drupal.

What about Blogdown?

We did our due diligence, and considered the possibility staying within the R Markdown ecosystem and working with something like Bookdown/Blogdown. There is a case to be made for working through a static site generator and avoiding the CMS altogether. For a single researcher or a small dedicated team that includes at least 1 person who enjoys being a part-time webmaster this approach is a good one.

At Urban, we have organizational requirements that come into play. We also have resources to support the core research work for which we are known. We don’t expect our researchers to be responsible for deploying accessible, search-engine friendly, semantically correct, browser-compliant, responsive code to the World Wide Web on a daily basis.

Also, we anticipate greater demand for publishing dynamic documents. We are at the beginning of a long and exciting journey. If Drupal is NOT the answer, well we needed to find that out sooner rather than later. We dove in. Let's dive in.

The Setup

As this is our first Drupal post, you may want to know a bit about our stack. We keep it simple. We use Pantheon, and we leverage the Terminus Build Tools Plugin to rapidly stand up a base Drupal installation, Github repository, Pantheon Sandbox site, and CircleCI testing and deployment workflow. The first Pantheon link back there explains it best. If you are determined to follow along, start there (we don't recommend trying to follow along).

The Design System

If you were to wring all of the tech and tools out of this blog post, you'd be left with little more than a single, over-arching thought: *We made this all work by creating a simple and well-documented design system*. That's pretty much it. Seriously.

There are plenty of resources online, if you are looking for a deep dive into design systems. Sarah Feldmans's Medium post is essential reading for those looking to embark on a design systems journey and Nathan Curtis's post is an excellent overview (with copious links and references). If you really want to get busy, checkout the repository of design systems from well-known brands.

Only a few paragraphs in and we've surely exceeded the record for "design" mentions in a Data Science blog. Mission. Accomplished.

For our purposes, the design system is the formal collection of components that, when combined, make a user interface. We build and refine those components using a tool called Pattern Lab. This allows us to quickly dive into our front-end build before we even have a Drupal site. It also provides a generated, user-friendly style guide.

To assist with turning our Pattern Lab bits into actual working Drupal components, we leverage the amazing Particle starter kit, an open source project from Phase2 Technology. Particle, by default, leverages the Bootstrap 4 front-end component library. Be sure to check out the Particle docs if you want to learn more. We can't recommend it enough.

Design system purists would correctly point out that we are really describing a "theme" here, rather than a true "Design System." The two concepts are related, but generally a design system encompasses more than a single product (think 20 Web sites, an app, and an email template), whereas a Website theme defines the look and feel of a particular instance.

Even though NCCS is a single Website, we needed to be able to communicate the site's style guide to research authors who generate R Markdown documents. This is not a minor detail. For R Markdown documents to be fully integrated into the parent site, they must utilize the parent site's stylesheet. This means a researcher or document author must generate their R Markdown files WITHOUT embedded styles.

We used a design system approach to standardize document styles and communicate those standards with online guides and living documents. More than any single technology innovation, the design system concept provided the necessary framework to bridge the gap between generated R Markdown documents and the parent site into which they are posted. **This** was the innovation – the interesting part.

The R Markdown

Preparing an R Markdown file for publication on NCCS is as simple as setting the output parameter in the document frontmatter:

Here's a link to a gist, which is clumsily recreated below

~~~~
```
output:
  github_document:
    html_preview: true
always_allow_html: yes
```

Example:

```
---
output:
~~~~

```
  github_document:
    html_preview: true
params:
  NCCSDataYr: 2015
always_allow_html: yes
---

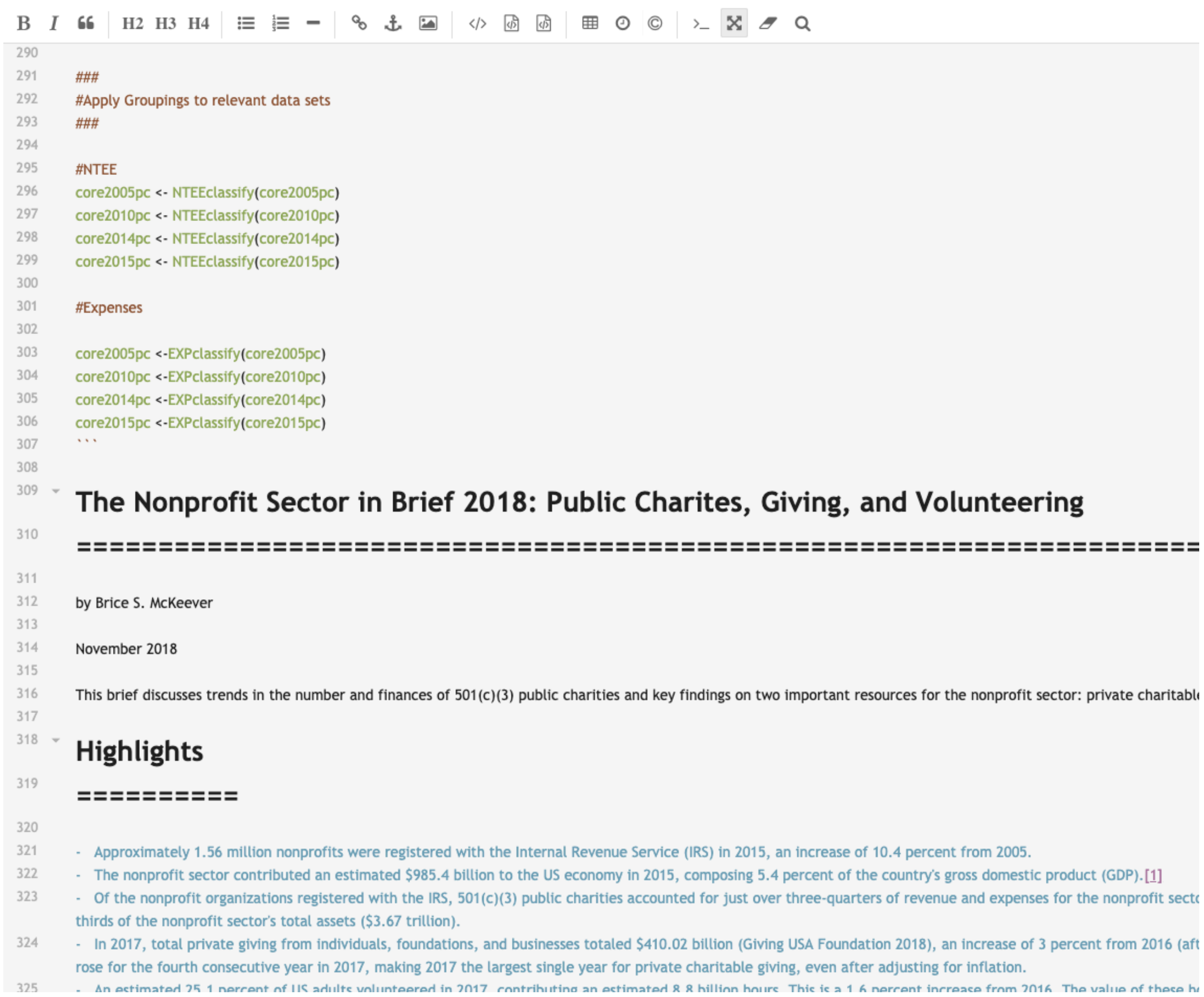
```
```

We use Github Document, with Github Flavored Markdown (GFM) for NCCS because it has enhanced support for table formatting and raw HTML (*always_allow_html: yes*). While a "pure" Markdown source file will always yield the most predictable display, HTML provides research authors with additional formatting tools to communicate important concepts. Need to highlight a specific table row? There's a documented class for that.

By default, Markdown output does not embed supporting files. Artifacts such as generated figures or supporting graphics are output to a configurable directory in the research author's project.

The Editor Experience

What is the point of going through all of this trouble if we end up adding friction on the edit screen? We put a good deal of effort into anticipating (and working to avoid) that question. The result is a fairly simple and (we hope) intuitive experience.
```



```
###
#Apply Groupings to relevant data sets
###

#NTEE
core2005pc <- NTEEclassify(core2005pc)
core2010pc <- NTEEclassify(core2010pc)
core2014pc <- NTEEclassify(core2014pc)
core2015pc <- NTEEclassify(core2015pc)

#Expenses

core2005pc <-EXPclassify(core2005pc)
core2010pc <-EXPclassify(core2010pc)
core2014pc <-EXPclassify(core2014pc)
core2015pc <-EXPclassify(core2015pc)
```

The Nonprofit Sector in Brief 2018: Public Charites, Giving, and Volunteering
===

by Brice S. McKeever

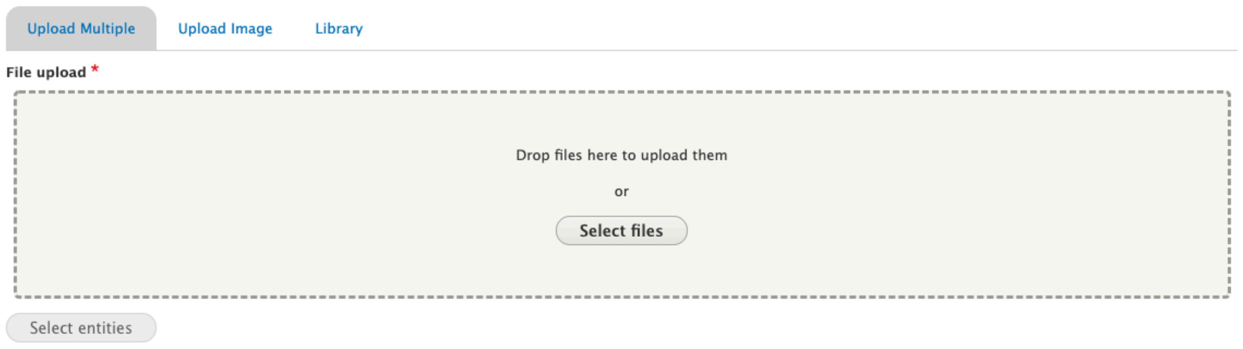
November 2018

This brief discusses trends in the number and finances of 501(c)(3) public charities and key findings on two important resources for the nonprofit sector: private charitabl

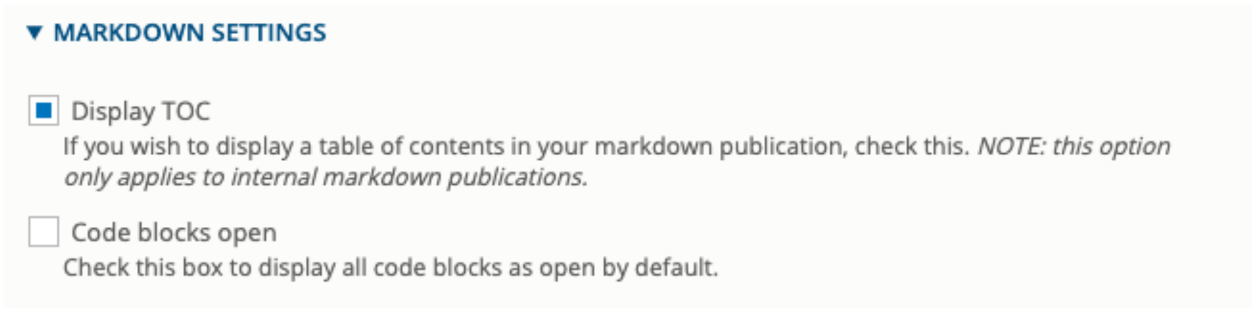
Highlights
==========

- Approximately 1.56 million nonprofits were registered with the Internal Revenue Service (IRS) in 2015, an increase of 10.4 percent from 2005.
- The nonprofit sector contributed an estimated $985.4 billion to the US economy in 2015, composing 5.4 percent of the country's gross domestic product (GDP).[1]
- Of the nonprofit organizations registered with the IRS, 501(c)(3) public charities accounted for just over three-quarters of revenue and expenses for the nonprofit secto
thirds of the nonprofit sector's total assets ($3.67 trillion).
- In 2017, total private giving from individuals, foundations, and businesses totaled $410.02 billion (Giving USA Foundation 2018), an increase of 3 percent from 2016 (aft
rose for the fourth consecutive year in 2017, making 2017 the largest single year for private charitable giving, even after adjusting for inflation.
- An estimated 25.1 percent of US adults volunteered in 2017, contributing an estimated 8.8 billion hours. This is a 1.6 percent increase from 2016. The value of these h
```

Editor.md provides the Markdown editor. Research authors paste their Markdown and can edit as needed.


```

*Supporting files are uploaded via drag and drop. The site will automatically place images where they belong in the document.*


▼ MARKDOWN SETTINGS

■ Display TOC
If you wish to display a table of contents in your markdown publication, check this. *NOTE: this option only applies to internal markdown publications.*

☐ Code blocks open
Check this box to display all code blocks as open by default.

*We provide a few basic formatting options so layout can be matched with content.*

## The Filter Plugins

The functional backbone of this project is the Drupal Filter module and, more specifically, its Filter API.  This set of tools provides a framework for evaluating and transforming text input. It leverages the Drupal 8 Plugin API, which is a beautiful thing. Because filters are swappable and pluggable, you can easily make your own. So we did just that.

But first, module shopping! In this age of open source, you don't get bonus points for re-creating existing tools or functionality. Quite the opposite. The less custom code you have to write to achieve your objective, the better. That's less code you need to maintain, secure, and document (because we all document perfectly all the time, don't we?). Worry not, we wrote plenty of code on this project. We just chose our moments strategically.

## The Interlude

A lot of brilliant people write a lot of brilliant software that most of us use and extend gratis. Try to support these people when and where you can. If you want to support the Drupal community, support the Drupal Association. /soapbox

## The Digression

Before we wrote any custom code (which we will talk about in this post, we promise), we researched, assembled, installed, configured, and tested a variety of contributed modules. Here are the modules we selected as our starting point.

Each of these modules provides a filter, which transforms text input in some specific way:

Markdown
Most of the actual markdown rendering is handled by this module, which provides a configurable text filter to handle the transformation from markdown to HTML. The project is in active development, and there is a major refactor in progress. If you are looking into using this module, be sure to read this issue. We are currently using version 8.x-2.0-alpha1.

GeSHi Filter
A Drupal implementation of the GeSHi Generic Syntax Highlighter library for PHP. It is a server-side solution that uses the Drupal Filter API to transform monochromatic code chunks into a reader-friendly experience. GeSHi supports 251 languages, including R.

TOC API/TOC Filter
Yet another contributed filter module, TOC API and TOC Filter provide a configurable framework for parsing html and generating a Table of Contents (TOC) menu from header tags (h1, h2, h3, etc). We use JavaScript to accomplish the same on some of our other Drupal sites. Both approaches work great and have their merits. In this case, the ability to control TOC markup by overriding a twig template won the day.

## The Custom Code

So, as mentioned above, we wrote some filters. And in this part I will be showing an example of the FilterAddCodeCollapse filter plugin. It parses the string and adds the necessary elements and classes to make all code blocks collapsible.

In this example we'll show:

- Generate plugin with drupal console
- Extend FilterBase class
- How to use DOMDocument to parse and replace
- How to make a filter configurable per node.