Lsci 202A Exercise 3

Prof. Futrell

This exercise will consist of making an academic website using what you have learned about (1) Github, (2) the command line, (3) text/code editors such as Atom and RStudio, and (4) a little bit of Python.

Some of you may already have an academic website. You may keep it, but you will still find this exercise useful, and you may want to integrate some of the parts of the website we build here into your already-existing website.

You will turn in a link to the Github repository containing your web content.

1 Getting started

1. First, we will need to install a piece of software called Hugo that will build your website. (You should already have `git` and a text editor such as Atom installed, and have made an account on Github.)

   (a) On macOS, run `brew install hugo` and `brew install golang`

   (b) On Windows, there are two steps:

      i. Follow the instructions at
      `https://sal.as/post/install-hugo-on-wsl/`
      You will need to download the *extended* Hugo release:
      `hugo_extended_0.79.0_Linux-64bit.deb`

      ii. Install Go version 1.15 or higher:
      ```
      wget https://golang.org/dl/go1.15.5.linux-amd64.tar.gz
      sudo tar -xz -C /usr/local -f go1.15.5.linux-amd64.tar.gz
      export PATH=$PATH:/usr/local/go/bin
      echo "export PATH=$PATH:/usr/local/go/bin" >> ~/.profile
      ```

2. Test that hugo is installed by running `hugo version` on the command line. If it works, this should print out some help information about Hugo.

3. Optional: If you would like to edit your text in RStudio (rather than Atom or something else) then you will need to open RStudio and run in the REPL `remotes::install_github('rstudio/blogdown')`

4. You should also install the Python package `academic`. To do so, run `pip install academic`

2 Publishing your site

You will build your website by modifying the files that you just cloned from Github. When you have modified your files and you want to publish them, you will do this by pushing your files into a special Github repository. Once you do this, your files will appear on a website with a URL like `http://yourname.github.io`

Here, will will start by publishing your website as-is, before we start making any changes to the files that define the content.

This part of the exercise involves the most technical command-line stuff. Once you complete this part successfully, you should not have to run any of this again: you will be able to deploy your website simply by running `git push`. But before we can do that, there is some setup to do:

1. Create two new repositories on Github:
   (a) First, using the 'create repository' feature, create a repository that will contain your published website, called `YOUR-GITHUB-USERNAME.github.io` You should initialize the repository with a README file.
   (b) Second, you will create a repository to contain the content of your website. To do this, go to the repository `https://github.com/wowchemy/starter-academic` in your browser and click "Fork". This will create a copy of the `starter-academic` repository under your own username which you can edit.
2. Navigate to the directory where you want to keep your website files, and clone the content repository with
   `git clone https://github.com/YOUR-GITHUB-USERNAME/starter-academic.git`
3. Test that Hugo is working by navigating into the cloned repository on the command line and running `hugo server`, and then navigating to `http://localhost:1313/` in your browser. You should see a welcome page.
4. On the command line, inside the cloned repository, run `git submodule update --init --recursive`
5. If there is a directory called `public` inside your content repository, remove it using `rm -rf public` (Be careful! This is the command that really deletes files!)
6. `git submodule add -f https://github.com/NAME/NAME.github.io public`
7. Now you are ready to push your local content repository to github. Run:
   ```
   git add .
   git commit -m "Initial commit"
   git push
   ```
8. Open your copy of the `starter-academic` repository as a project in your text editor (such as Atom).

9. Open the file
   `config/_default/config.toml`
   and set
   `baseurl = "https://YOUR-GITHUB-USERNAME.github.io/"`

10. Now we are finally ready to push the website to the web. First, run
    `hugo`
    When the `hugo` command runs, it generates a bunch of files and dumps them in the `public/` folder. These files comprise your public-facing website. Now, add the contents of `public` and push them:
    ```
    cd public
    git add .
    git commit -m "Build website"
    git push
    cd ..
    ```

11. Now navigate to `https://USERNAME.github.io` in your browser. Your website (currently just a template) should be there!

How to publish your changes to the web.

1. Navigate into to your `starter-academic` repository
2. Run `hugo`
3. Navigate into your `public` directory with `cd public`

4.
   ```
   git add .
   git commit -m "Build website"
   git push
   ```

3 Setting parameters

Hugo is what is called a static website generator. The way it works is that there are a bunch of text files which determine the content of your website; these text files are easy for humans to edit and understand. When you run the command `hugo`, these text files are processed into the HTML that defines the website which will actually be displayed in a browser. Now, we will look at how these text files are structured, and how you can edit them to make a useful website.

1. **Choosing a visual theme.** In the file
   `config/_default/params.toml`
   you will see a field called `theme`. Above it there is a link to the different color themes you can use. Try out some different themes and see what you like!

   (a) To try something out, the best thing to do is to edit a file, save it, then run `hugo server` and navigate to `http://localhost:1313/` to see what it looks like. Until you push to Github, none of your changes are public.

2. **Updating information** There are a number of other fields in `config/_default/params.toml` that you should update. Go through these and update them. If you do not wish to specify them, set them equal to an empty string, “”.

   (a) `org_name`: Department of Language Science, University of California, Irvine

   (b) `description`: A short description of yourself, like “Graduate student in Language Science at UC Irvine”

   (c) `email`, `phone`, `address`, `directions`, `office_hours` (you'll likely want to update these each quarter), `contact_links`, `twitter`.

Once you have set these parameters, try running `hugo server` and looking at the resulting website. Check that everything is as you want.

4 Content

Now we will populate your website with real content. After each step, you should check how the website looks by running `hugo server`.

The content for your website lives in the `content` directory.

When you are writing text for your website, if you want to include formatting or links, use Markdown syntax:
`https://www.markdownguide.org/basic-syntax/`

The first thing we will need to do is input your personal information (name, position, a picture, etc.). To do this, we need to change the file `content/authors/admin/_index.md`.

1. **Choosing what to display**. Hugo defines the content of a website in terms of a sequence of 'widgets' (for example, the part that displays your personal information is a widget, and the part that lists your publications is another widget). Each widget is defined by a file in the folder `content/home/`.

   (a) The first thing we want to do is to choose which widgets get displayed. To do this, open the file for each widget in your text editor, and set `active: true` for the ones you want, and `active: false` for the ones you don't. (Inside some of the files it will be `active = false`; check to see how the parameters are set inside each file.) I recommend setting `about`, `contact`, and `publications` to `true`, and the others to `false`. But feel free to play around!

   (b) You might also want to change the order that widgets are displayed in. This is controlled by the `weight` parameter in the `.md` files.

   (c) To change the menu on top, modify the file `config/_default/menus.toml` This is also where you can add a link to a PDF of your CV.

2. **Adding your personal information**

   (a) Open `content/authors/admin/_index.md` in your text editor
   (b) In the `title` field, add your full name.
   (c) In the `role` field, add your position (e.g., PhD Candidate, Graduate Student, etc.)
   (d) In the `bio` field, write a brief sentence to describe yourself.
   (e) Add the `organizations` you are affiliated with.
   (f) Fill out the rest of the fields. This should be relatively easy to do.
   (g) At the bottom, write a slightly longer bio of yourself.
   (h) Move a picture of yourself into the `content/authors/admin` folder, naming it `avatar.jpg` or `avatar.png`
   (i) Now when you look at the website, it should have your information and picture at the top!

3. **Publications**. The publications section is populated using files in the `content/publication/` folder. Let's add the files you need to display your publications. These could be publications in journals or conference proceedings, or they could be your various theses or manuscripts that you would like to share, even if they are not officially published.

   (a) The best way to manage your publications is in something called a `bibtex` file. If you use something like Zotero, you can export a Bibtex file from Zotero. Otherwise, you can create the file on your own in a text editor:
   (b) Create a file called `publications.bib` in a text editor. A `.bib` file consists of a series of entries, one per publication. You can get information on how to write bibtex entries here:
   `https://www.economics.utoronto.ca/osborne/latex/BIBTEX.HTM`
   (c) Move this `.bib` file into the root directory for your website.
   (d) Run `academic import --bibtex your-file.bib`
   This will populate the `content/publication` directory with folders, one per publication.
   (e) Go into each folder and look at the files named `index.md`. Inside each one, you can set `featured` to `true` to make sure that this publication is displayed on the main page.
   (f) Inside each `index.md` file, you can also provide a field `url_pdf: http://...` with a link to the pdf file for that publication. You can also put a pdf file directly in the directory with the `index.md` file, and then link to it with `url_pdf: papers/your_pdf.pdf`
   (g) Inside each directory, you can add a `featured.png` file containing a striking image from the publication. This will be displayed if you set `view` to `3` in `content/home/publications.md`

Now view the results and push them to Github to publish them. Congratulations, you now have an informative and stylish academic website!