Setting up a blog

John Tipton

`blogdown` and gitHub

- Setting up a blog is easy using the `R` package blogdown
- Create a gitHub account
  - gitHub will be your public code repository
  - gitHub will be an advertisement for your skills, knowledge, and abilities as a data scientist

New server

- http://statistical-methods.ddns.uark.edu/
  - username: your Uark username
  - password: your Uark username
  - This server is not connected to the University system. There is no private or personal data on the server beyond what you upload. **Do not upload any personal or sensitive information to this server!**

Changing your server password

- Once you have logged into the server, open up a terminal window and type `passwd`. Then type your current password (no text will appear to protect your privacy), then choose a new password and type it twice. **Please do not use your university password -- make a new one.**
- This password does not need to be very secure as the only data will be for this project.

Setting up git on the server

- Make sure the `blogdown` and `usethis` packages are installed on the server

```
install.packages("usethis")
install.packages("blogdown")
```

- These instructions follow those at https://happygitwithr.com/ -- chapters 7, 9, and 11
- In the terminal, type

```
git config --global user.name 'Jane Doe'
git config --global user.email 'jane@example.com'
git config --global --list
```

where the `user.name` 'Jane Doe' is your gitHub username and the `user.email` 'jane@example.com' is your gitHub email.

Setting up a gitHub SSH key

- In the terminal, type

```
ssh-keygen -t rsa -b 4096 -C "USEFUL-COMMENT"
```

where you can change the text in "USEFUL-COMMENT" to describe what the key is. I use "statistcal-methods-server" to distinguish this from my "home-desktop" and "work-desktop" computers.

- Accept the proposal to save the key in the default location. Just press Enter here:

```
Enter file in which to save the key (/Users/jrtipton/.ssh/id_rsa):
```

- Next, you have the option to enter a passphrase for extra security. For now, skip this but for future work that might be sensitive, an additional passphrase can be useful (and can be saved in a password manager)

Setting up a gitHub SSH key

- The completed process should look something like:

```
jrtipton@statistical-methods ~ $ ssh-keygen -t rsa -b 4096 -C "statistical-methods-server"
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/jrtipton/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /Users/jrtipton/.ssh/id_rsa.
Your public key has been saved in /Users/jrtipton/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:ki0TNHm8qIvpH7/c0qQmdv2xxhYHCwlpn3+rVhKVeDo USEFUL-COMMENT
The key's randomart image is:
+---[RSA 4096]----+
|      o+   . .   |
|     .=.o . +    |
|     ..= + +     |
|      .+* E      |
|     .= So =     |
|    .  +. = +    |
|   o.. = ..* .   |
|  o ++=.o =o.    |
| ..o.++o.=+.     |
+----[SHA256]-----+
```

Setting up a gitHub SSH key

- Adding the ssh key to the agent
- Make sure the ssh agent is running by typing in the terminal:

```
eval "$(ssh-agent -s)"
```

which should return something like:

```
Agent pid 59566
```

- Then, add your SSH key in the terminal using

```
ssh-add ~/.ssh/id_rsa
```

Setting up a gitHub SSH key

- get out your SSH key using RStudio
- Go to Tools > Global Options…> Git/SVN. If your key pair has the usual name, id_rsa.pub and id_rsa, RStudio will see it and offer to “View public key”. Do that and accept the offer to copy to your clipboard. If your key pair is named differently, use another method.
- Highlight and copy the text of the key making sure to copy the key starting with `ssh-rsa`
- If there is no "View public key" option, you can go to the terminal and type

```
cat ~/.ssh/id_rsa
```

- copy the **entire output** (starting with ssh-rsa to the end) to your clipboard (ctrl-c) or using your mouse (right click)

Setting up a gitHub SSH key

- Make sure you're signed into GitHub. Click on your profile pic in upper right corner and go Settings, then SSH and GPG keys. Click "New SSH key". Paste your public key in the "Key" box. Give it an informative title, presumably related to the comment you used above, during key creation. For example, you might use statistical-methods. Click "Add SSH key".

In theory, we're done! On the server you can type in the terminal

```
ssh -T git@github.com
```

to test your connection to GitHub (answer **yes** to continue). If you're not sure what to make of the output, see the link for details. Of course, the best test is to work through the realistic usage examples elsewhere in this guide.

Setting up your gitHub PAT

To access your gitHub repository, you will need a personal access token (PAT) -- see here for details about how to set this up as this is what I based the following tutorial on.

- Log into the class server and install the `usethis` package

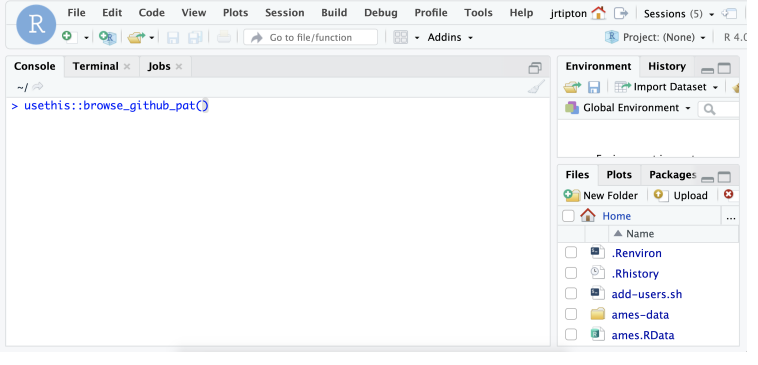
```
install.packages("usethis")
```

Setting up your gitHub PAT

Once the `usethis` package is installed, type

```
usethis::browse_github_pat()
```

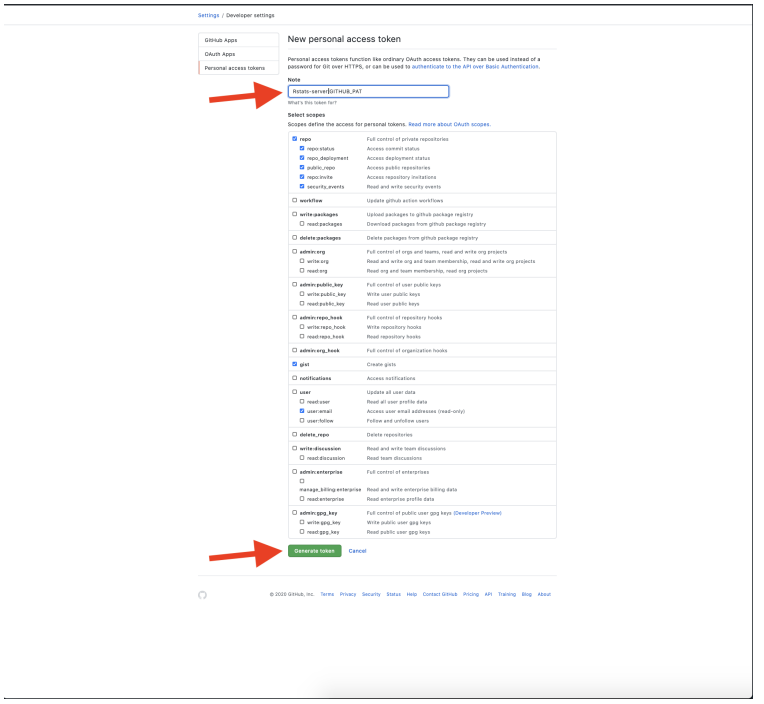
to open a webpage using your gitHub account (you will likely need to enter your password to continue).



This will open up a webpage.

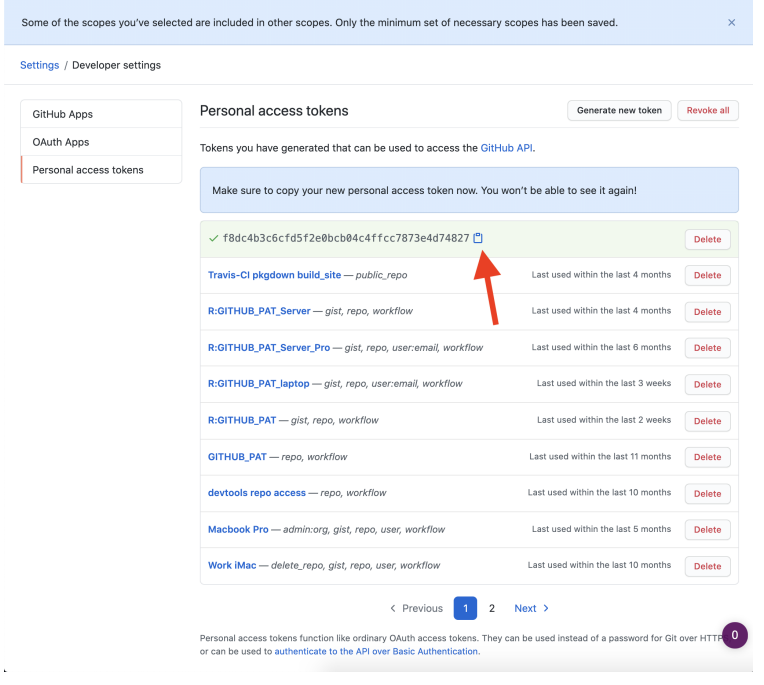
Setting up your gitHub PAT

On this webpage is a form to create your PAT with reasonable settings. Give the PAT a nickname and click "Generate token" and the token will be displayed.



Setting up your gitHub PAT

The token is a string of 40 random letters and digits. Make sure you **copy this token to your clipboard** as this is the last time you will be able to see it. You can copy by clicking on the clipboard symbol.



Setting up your gitHub PAT

Once you have generated a gitHub PAT and copied it to your clipboard, we will add the PAT to your `.Renviron` file. The goal is to add the following line in your `.Renviron` file:

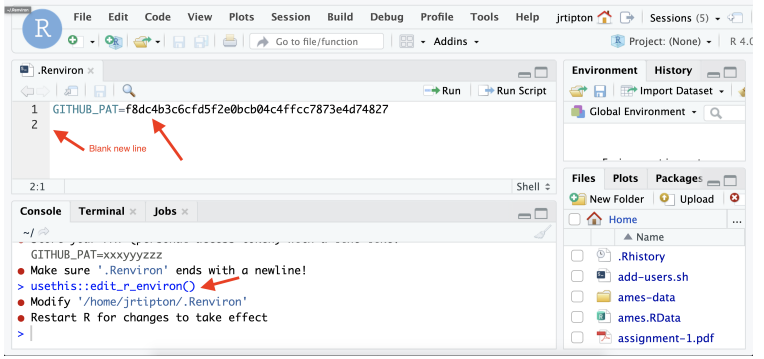
```
GITHUB_PAT=XXXXX
```

where the XXXX is the PAT copied from github. The `.Renviron` file is a hidden file that lives in your home directory and contains variables for `R` to load on startup.

Setting up your gitHub PAT

The `.Renviron` file can be edited in `R` using the `usethis` package. In `R` type

```
usethis::edit_r_environ()
```



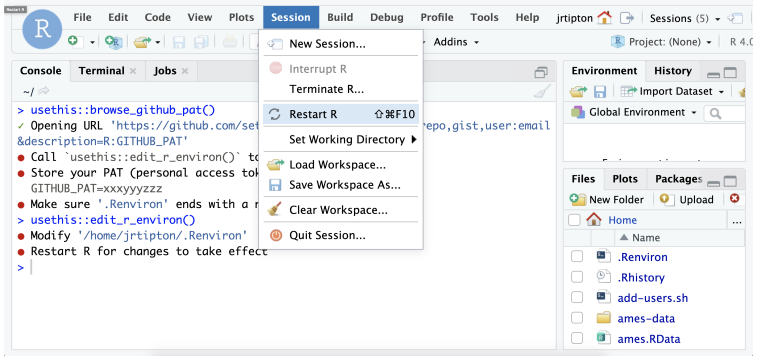
Your .Renviron file should pop up in your editor. Add your GITHUB_PAT as above,

```
GITHUB_PAT=XXX
```

where the `XXXX` is the PAT copied from the GitHub site with a line break at the end of the file save the `.Renviron` file and close it. If questioned, YES you do want to use a filename that begins with a dot `.`. Note that, by default, most dotfiles are hidden in the RStudio file browser, but `.Renviron` should always be visible.

Setting up your gitHub PAT

Restart `R` (Session > Restart R in the RStudio menu bar), as environment variables are loaded from `.Renviron` only at the start of an `R` session.



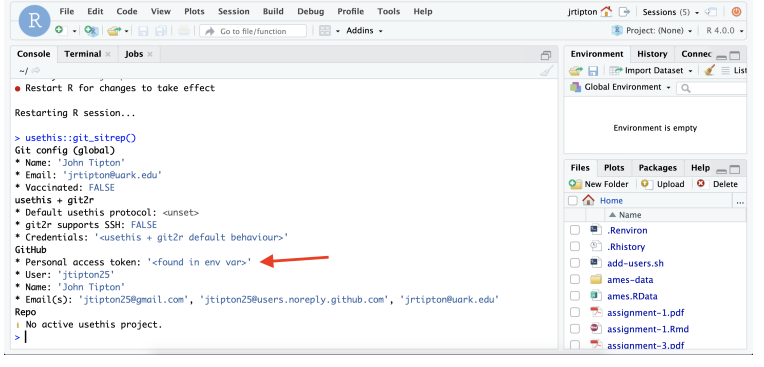
Setting up your gitHub PAT

Check that the PAT is now available like so:

```
usethis::git_sitrep()
```

You should see the following line in the output:

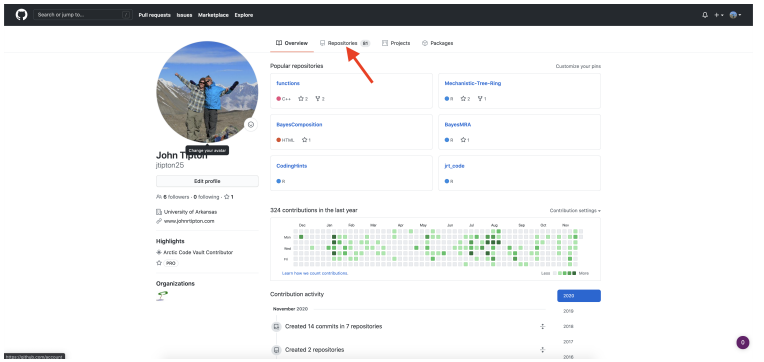
```
Personal access token: '<found in env var>'
```



Now commands you run from the terminal and from RStudio, which consults GITHUB_PAT by default, will be able to access GitHub repositories which you have access to.

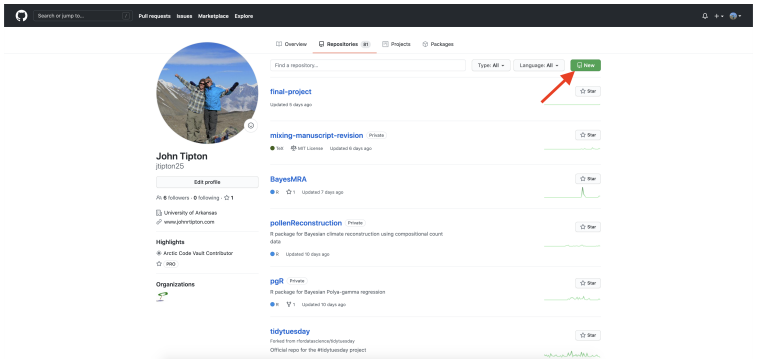
Creating a repository on gitHub

- Go to your gitHub webpage



Creating a repository on gitHub

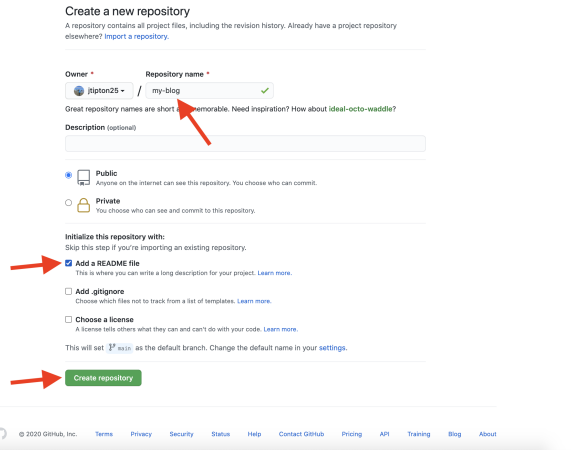
- click the Repositories tab



- click the green button to create a new repository

Creating a repository on gitHub

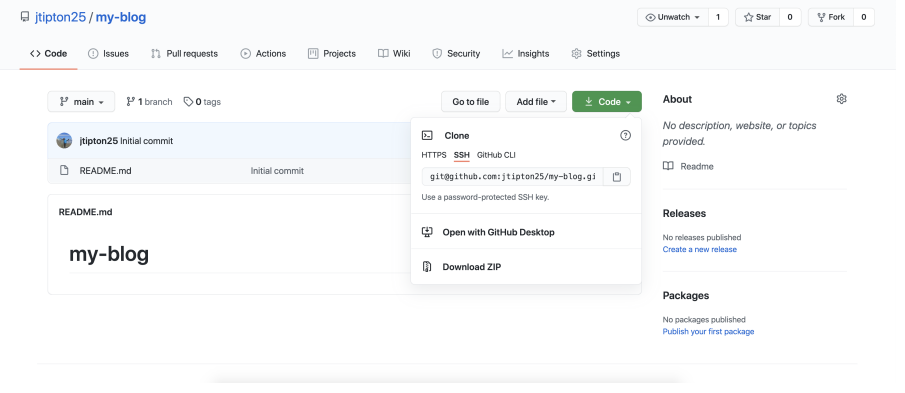
- Give the repository a name
- If you want, add a README to the repository
- Click the green "create repository button"



Cloning your gitHub repository

Once you have setup your PAT and created a repository on gitHub, you can clone your created repository to the server.

Open up the gitHub webpage for your repository and click on the green button that says XXX. Make sure you have the SSH tab highlighted and click on the clipboard symbol to copy the address into your clipboard.



Open up the terminal and type

```
git clone https://github.com/jtipton25/my-blog.git
```

where the exact site will depend on your gitHub username and repository name.

Creating the project

- Now that you have cloned the repository from gitHub, we will create a project.
- First, we make sure the `blogdown` library is installed (`install.packages("blogdown")`).
- Next we create a blog using

```
blogdown::new_site(dir = "~/my-blog")
```

where "~/my-blog" is the filepath for the repository you cloned from gitHub.

- Congratulations! You have just created your first blog!

Pushing the blog to gitHub

- open up the terminal and navigate to the directory your gitHub repository was cloned into

```
cd ~/my-blog
```

- check the status of the repository with

```
git status
```

- My status at this point is:

```
On branch main
Your branch is up-to-date with 'origin/main'.
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        .gitignore
        config.toml
        content/
        index.Rmd
        my-blog.Rproj
        static/
        themes/

nothing added to commit but untracked files present (use "git add" to track)
```

Pushing the blog to gitHub

- Add all the files with

```
git add .
```

- Commit all of the files (`-a`) with a message (`-m` to gitHub with

```
git commit -a -m 'first blog post'
```

- Push the files to the origin master branch with

```
git push
```

Git push error (Troubleshooting)

- See https://happygitwithr.com/ssh-keys.html#ssh-troubleshooting and https://docs.github.com/en/free-pro-team@latest/github/using-git/changing-a-remotes-url

If you get an error in the git push like this:

```
error: cannot run rpostback-askpass: No such file or directory
Username for 'https://github.com':
```

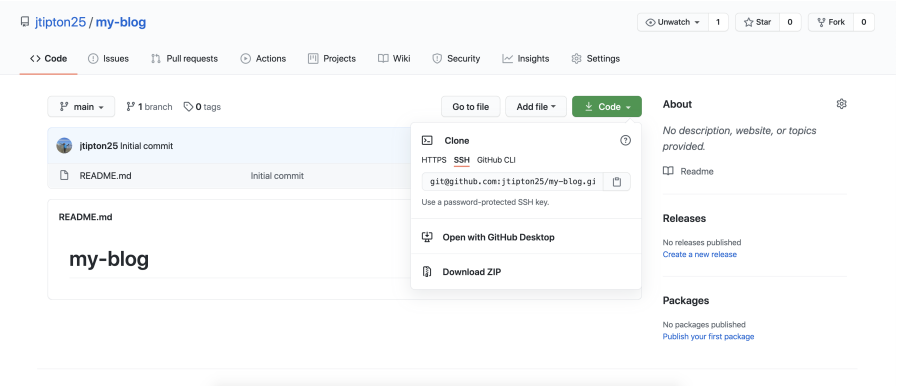
type `ctrl-c` to exit and check the remote using the terminal with

```
git remote -v
```

- There are two types of remotes (HTTPS and SSH), we want a SSH remote.
  - HTTPS remotes: `https://github.com/USERNAME/REPOSITORY.git`
  - SSH remotes: `git@github.com:USERNAME/REPOSITORY.git`

Git push error (Troubleshooting)

- If your remote is an HTTPS remote, change it to a SSH remote by going to the gitHub webpage and copying the SSH address (see Cloning your gitHub repository)



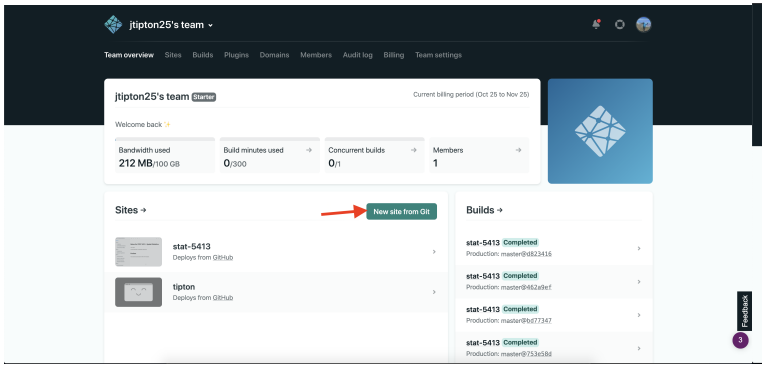
Then, change the remote URL to SSH using

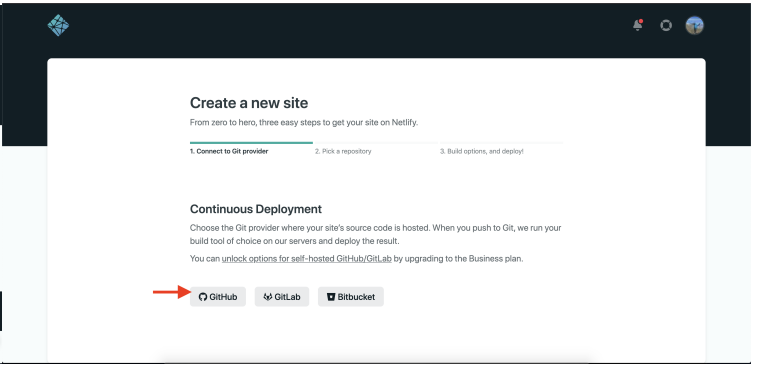
```
git remote set-url origin git@github.com:jtipton25/my-blog.git
```

where `git@github.com:jtipton25/my-blog.git` is changed to what your SSH remote is.

Setting up your Netlify account

- This section follows the instructions here
- Go to https://www.netlify.com/ and sign up for a free account using your gitHub account
- Follow the screenshots and instructions to link your gitHub account to Netlify
- Create a new site from git (green button in left figure)

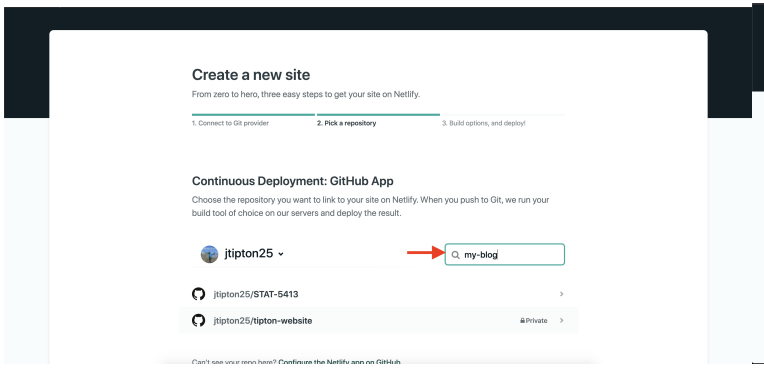


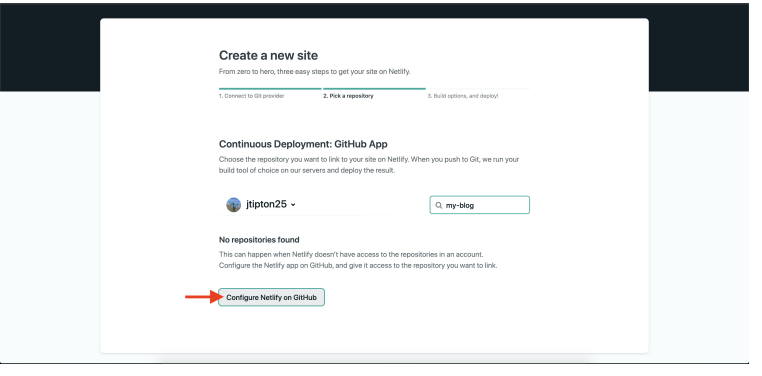


- Then choose to link to your gitHub account (gitHub button in right figure)

Setting up your Netlify account

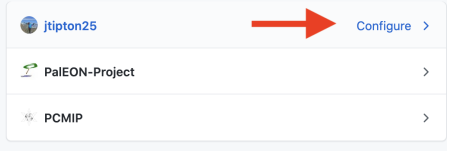
- Search for your gitHub repository (left figure)
- If you can't find your repository, click the configure repository button (right figure)



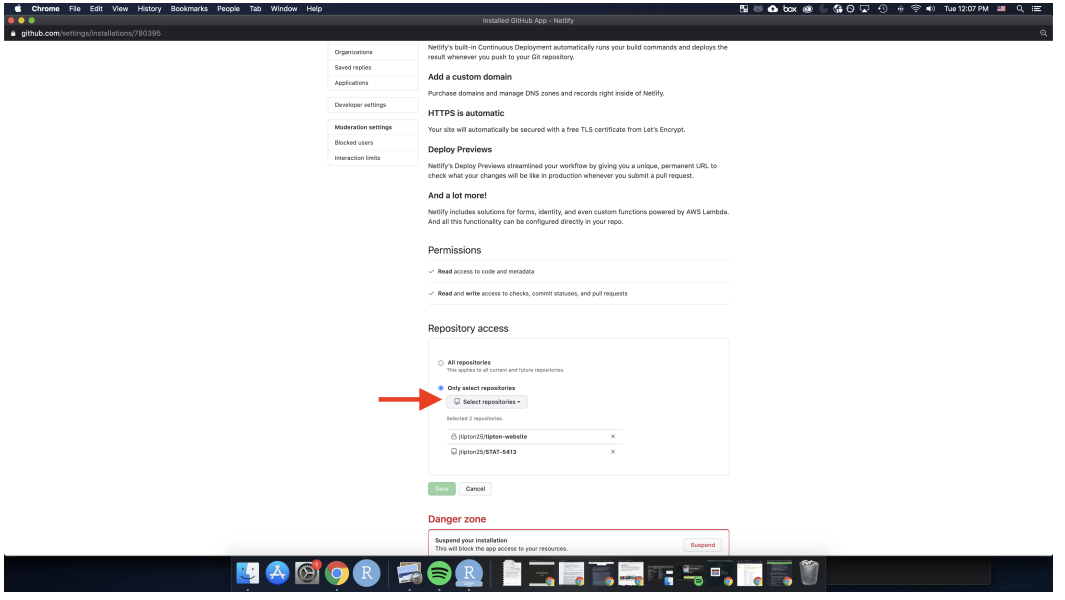


Setting up your Netlify account

- If you have multiple gitHub accounts/teams, you have to choose your correct gitHub account

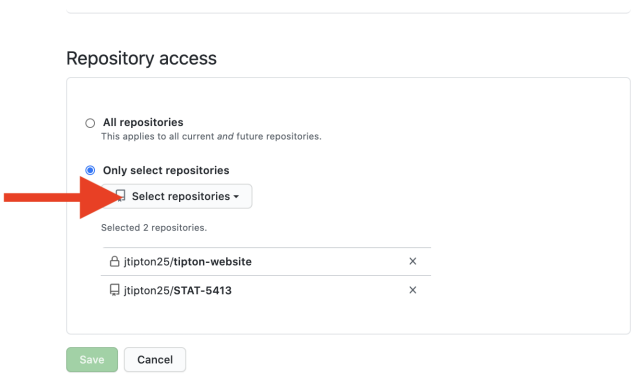


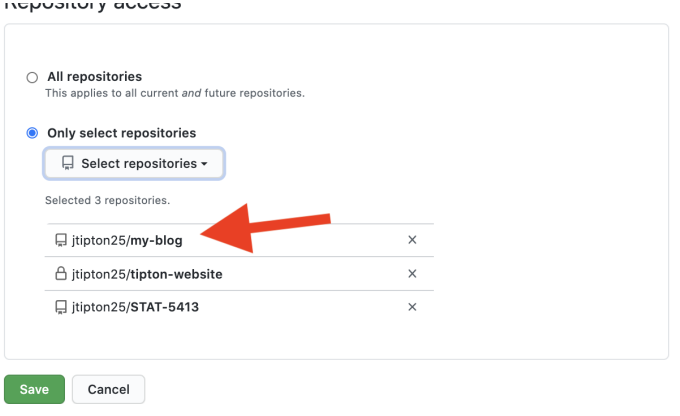
- I have chosen to only link certain repositories with Netlify. If you choose this option, click on the button to choose those repositories. Otherwise, you can share all your repositories with Netlify



Setting up your Netlify account

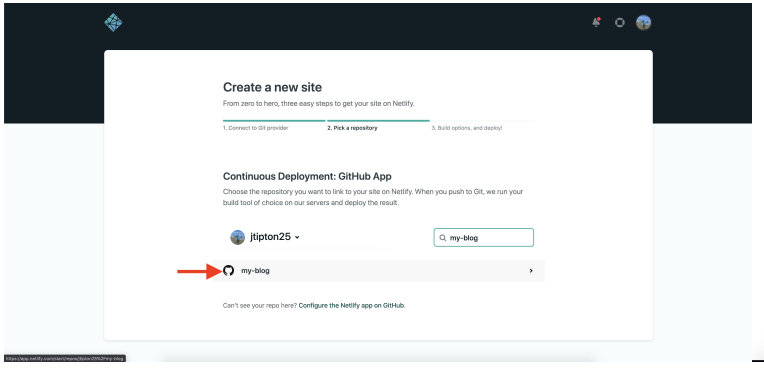
- If choosing to share only specific repositories, choose those repositories here. Make sure you choose the blog repository we just created.

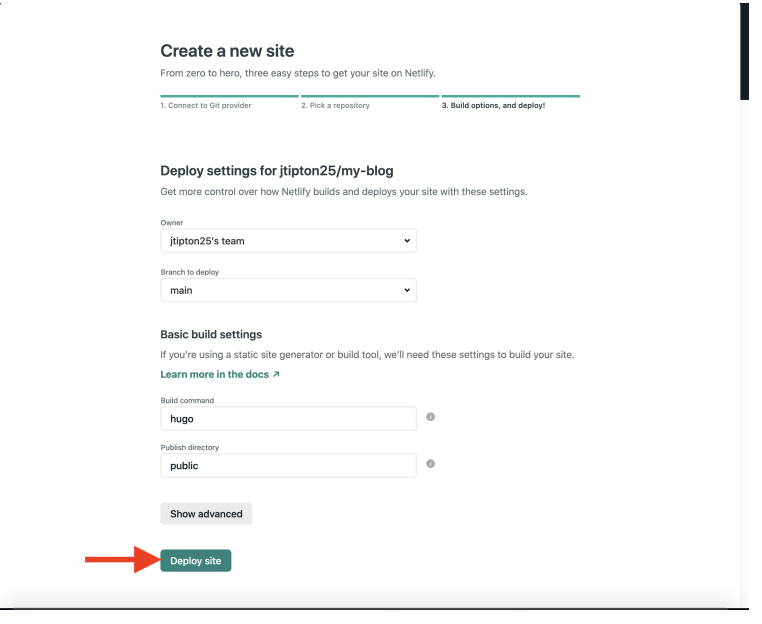




Setting up your Netlify account

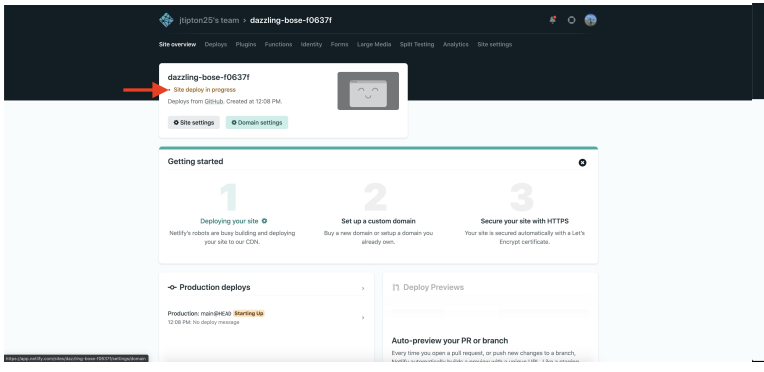
- Once you have chosen your linked repositories, you can choose on Netlify which repositories to build into a website (left figure). Once you have chosen the website you want to build, you can deploy the site (right figure)

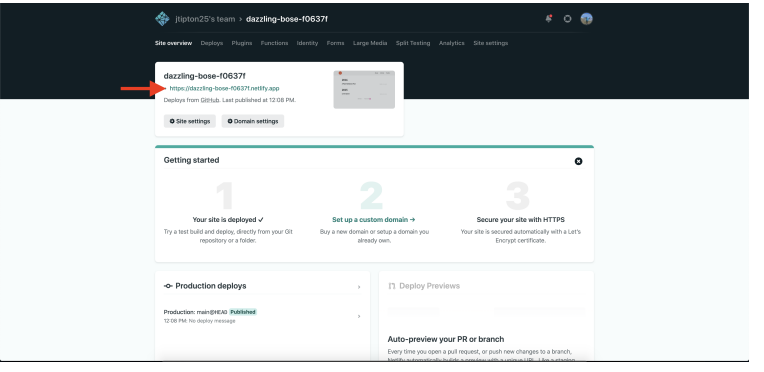




Setting up your Netlify account

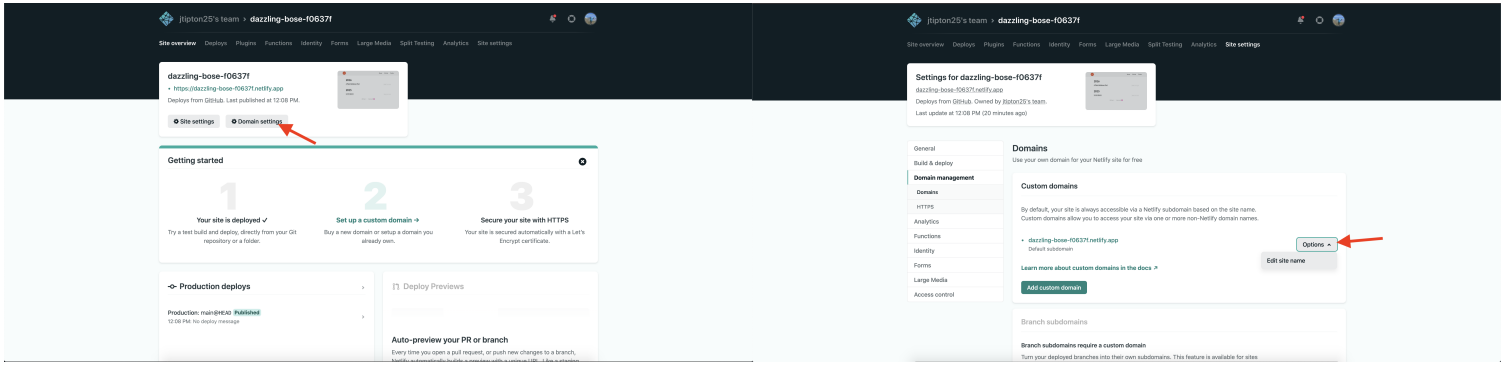
- This will take you to your deployed site (left). Once the site has built, you can view the site link using the link at the top (right)



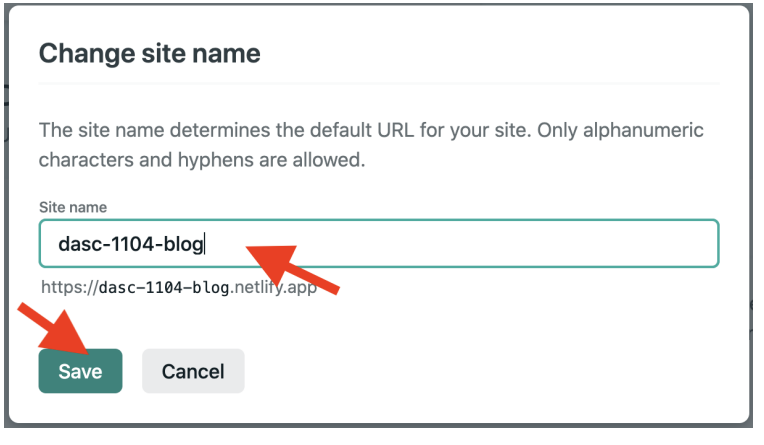


Setting up your Netlify account

- To change the blog address, we can edit the domain. Click on the Domain settings button (left). Then click the options button and edit site name.

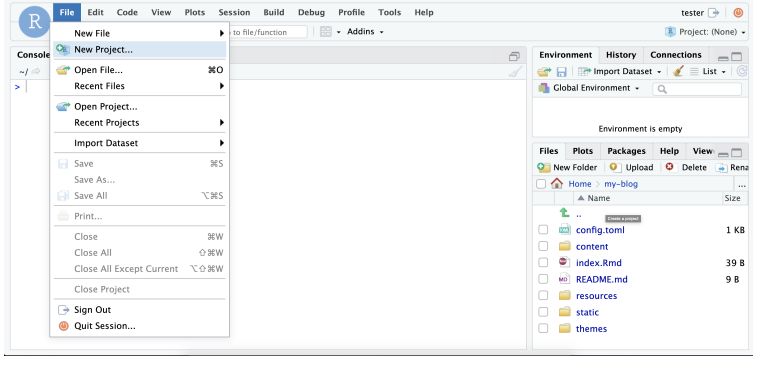


- Change the site name to something that you prefer



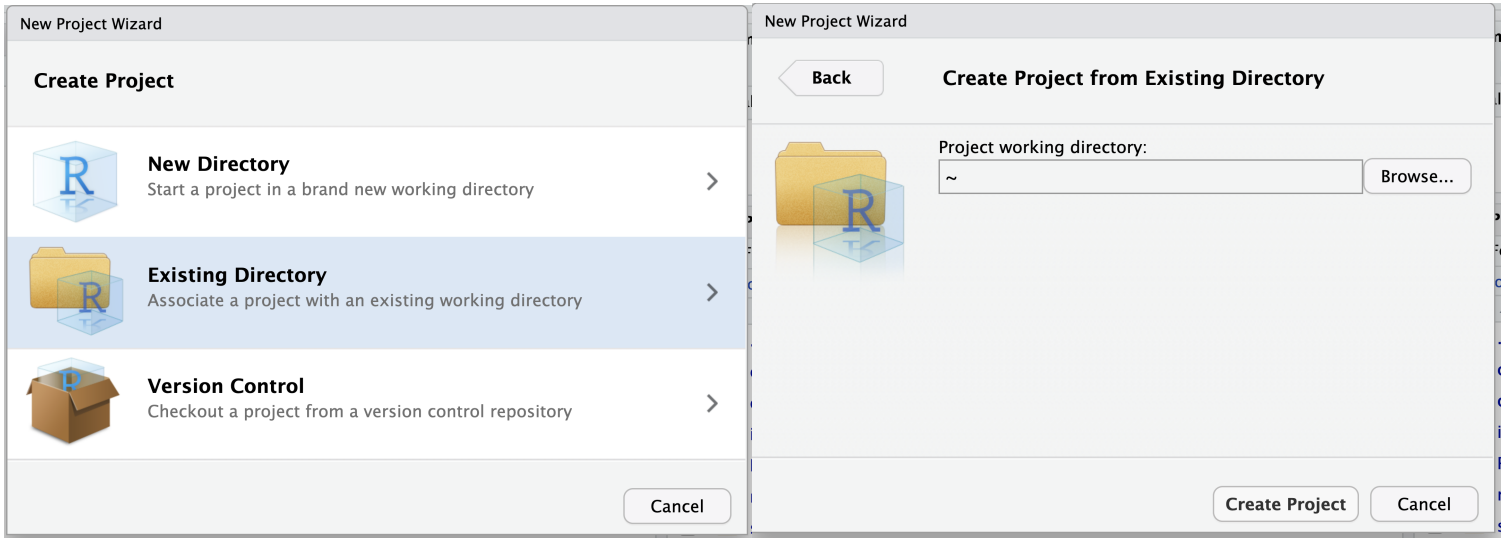
Editing the blog

- Awesome! We have a live blog. Now, all you have to do is make changes on the server, push them to gitHub, and then wait for Netlify to deploy the changes for your blog to automatically update.
- Adding a `.Rproj` file to the blog
  - In Rstudio, go "File - New Project". If prompted, don't save the RData.
  -



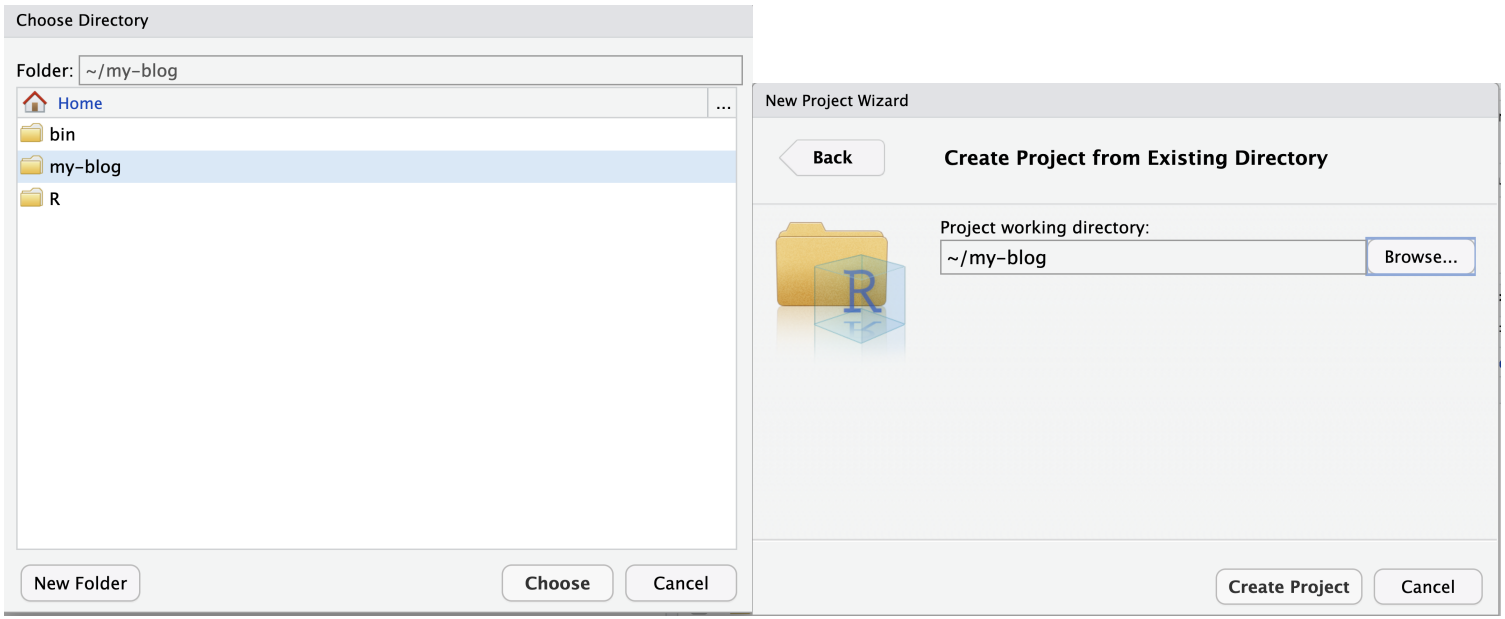
Editing the blog

- Choose an existing directory (left) and click browse to find the cloned repository (right)



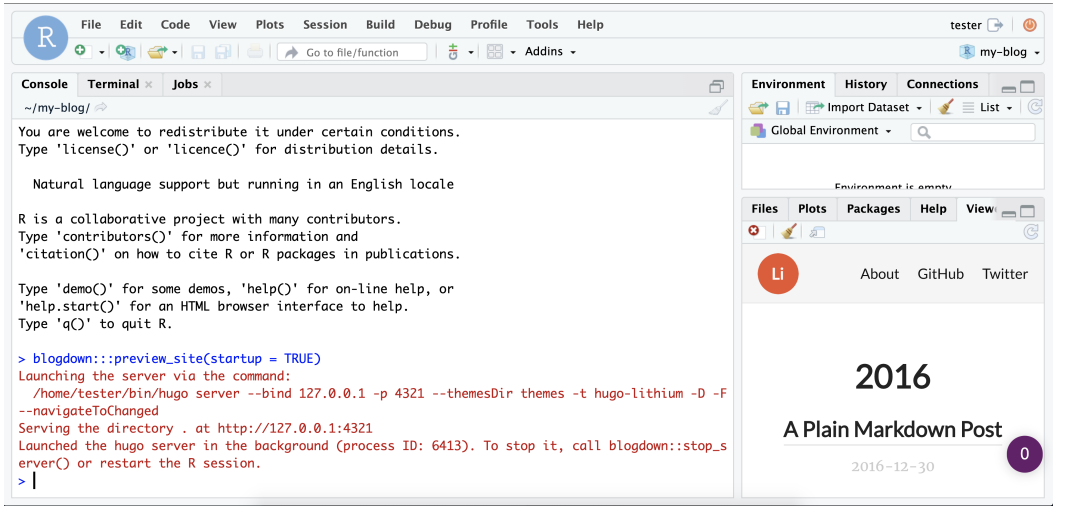
Editing the blog

- Choose the cloned repository folder (left) and create the project in this repository (right)



Editing the blog

- RStudio should then start a new session with your blog project loaded



Editing the blog

- Changing the site configuration
- The `config.toml` file changes global settings for your site. Let's open this file in `RStudio` and change the `title`, gitHub `url` and, if you want, your Twitter `url` (**Make sure these accounts are professional if you link to them!!!**)
- The `content` folder has a file `about.md`. Edit this file to tell us a little about yourself.

Editing the blog

- You can update the site locally (i.e., on the server, your own computer, but not on the website) using `blogdown::serve_site()`
    - This allows "real-time" previewing of your edits
- You can create a new post using `blogdown::new_post(title = "A first post")`
    - These posts live in the `content/post/` directory so you can edit them there (or delete the default posts)
- Before you push to gitHub to deploy your site, you can use `blogdown::build_site()` then push the repository to gitHub to update the site.
- More details are available at https://bookdown.org/yihui/blogdown/

Adding your resume to the blog

- As this is a new server, we need to install the `tinytex` package for converting a `.Rmd` file to a pdf. Type

```
install.packages("tinytex")
```

then

```
tinytex::install_tinytex()
```

to get this installed

- Then, copy the files from your resume repository (earlier in the semester -- particularly the files `resume-example.Rmd`, `svm-latex-resume.tex`, and `rick-martel-crop.jpg` -- or any images you added as well) to the `static` folder in your blog directory.

When I tried to compile the `.Rmd` file to pdf on the server, I got a compile error. To fix this, I ran

```
rmarkdown::render("./static/resume-master/resume-example.Rmd")
```

where the file path was where I uploaded the `.Rmd` file for the resume.

Adding your resume to the blog

- Next, you need to add the following to your `config.toml` file where the path of the url is the same path as where the `resume-example.Rmd` file was (without the `/static/` directory but with a leading "\"):

```
[[menu.main]]
    name = "Resume"
    url = "/resume-master/resume-example.pdf"
```

- Create a folder named `R` and add the file `build.R` that contains the following lines:

```
blogdown::build_dir('static')
```

- This file will take the files you added to the `static` directory (your resume) and build these files for hosting on the website when you run `blogdown::build_site()`

Adding your resume to the blog

- Now, run

```
blogdown::build_site()
```

to build the pdf document and

```
blogdown::serve_site()
```

to allow for interactive editing and to have your resume added to the blog

- To see your changes updated on the website, commit and push the gitHub repository.

Adding data

- If you add data for analysis (`.csv` files), you need to tell `blogdown` to ignore the `.csv` files. To do this, open the `config.toml` file and modify the line:

```
ignoreFiles = ["\\.Rmd$", "\\.Rmarkdown$", "_cache$", "\\.knit\\.md$", "\\.utf8\\.md$"]
```

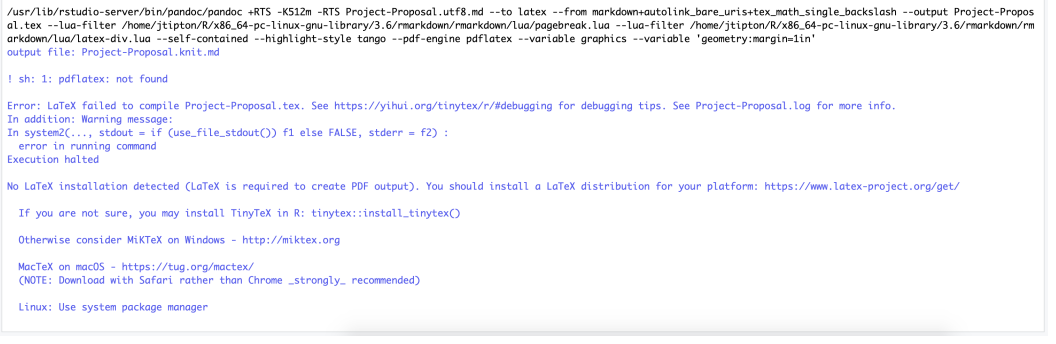
by adding `"\\.csv$"` to tell `blogdown` to ignore `.csv` files

so that the `ignoreFiles` line is now

```
ignoreFiles = ["\\.Rmd$", "\\.Rmarkdown$", "_cache$", "\\.knit\\.md$", "\\.utf8\\.md$", "\\.csv$"]
```

Troubleshooting: Knitting to pdf fails

- If you get the error shown below when knitting a `.Rmd` to `.pdf`, you don't have `tinytex` installed.



```
/usr/lib/rstudio-server/bin/pandoc/pandoc +RTS -K512m -RTS Project-Proposal.utf8.md --to latex --from markdown+autolink_bare_uris+tex_math_single_backslash --output Project-Propos
al.tex --lua-filter /home/jtipton/R/x86_64-pc-linux-gnu-library/3.6/rmarkdown/rmarkdown/lua/pagebreak.lua --lua-filter /home/jtipton/R/x86_64-pc-linux-gnu-library/3.6/rmarkdown/rm
arkdown/lua/latex-div.lua --self-contained --highlight-style tango --pdf-engine pdflatex --variable graphics --variable 'geometry:margin=1in'
output file: Project-Proposal.knit.md

! sh: 1: pdflatex: not found

Error: LaTeX failed to compile Project-Proposal.tex. See https://yihui.org/tinytex/r/#debugging for debugging tips. See Project-Proposal.log for more info.
In addition: Warning message:
In system2(..., stdout = if (use_file_stdout()) f1 else FALSE, stderr = f2) :
  error in running command
Execution halted

No LaTeX installation detected (LaTeX is required to create PDF output). You should install a LaTeX distribution for your platform: https://www.latex-project.org/get/

  If you are not sure, you may install TinyTeX in R: tinytex::install_tinytex()

  Otherwise consider MiKTeX on Windows - http://miktex.org

  MacTeX on macOS - https://tug.org/mactex/
  (NOTE: Download with Safari rather than Chrome _strongly_ recommended)

  Linux: Use system package manager
```

Run the following to resolve this issue

```
install.packages("tinytex")
tinytex::install_tinytex()
```

Troubleshooting: Knitting to pdf fails

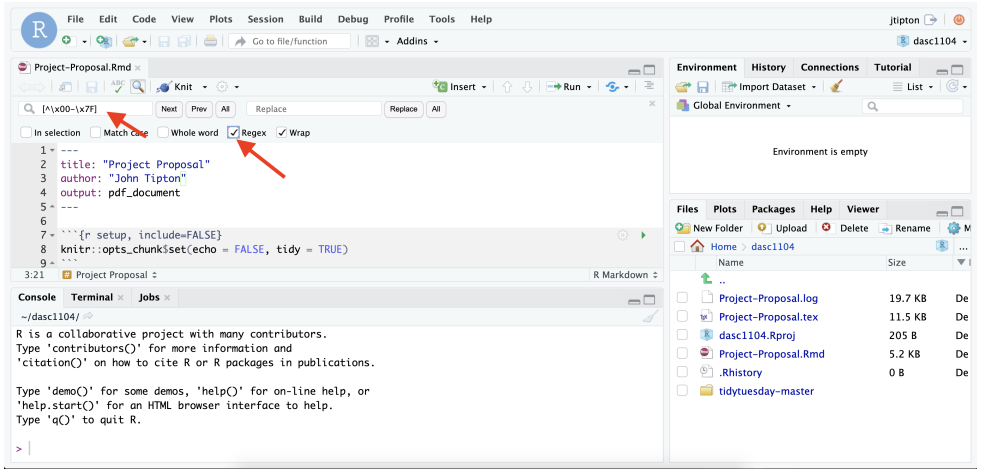
If you get the error shown below when knitting a `.Rmd` to `.pdf`, you have a unicode character in your `.Rmd`

```
output file: Project-Proposal.knit.md

! Package inputenc Error: Unicode character μ (U+03BC)
(inputenc)                not set up for use with LaTeX.

Try other LaTeX engines instead (e.g., xelatex) if you are using pdflatex. See https://bookdo
wn.org/yihui/rmarkdown-cookbook/latex-unicode.html
Error: LaTeX failed to compile Project-Proposal.tex. See https://yihui.org/tinytex/r/#debuggi
ng for debugging tips. See Project-Proposal.log for more info.
Execution halted
```

To find the unicode character(s), open the find bar using `ctrl-f` (`cmd-f` on Mac) or using the menu bar: `edit -> find`. Make sure the `Regex` box is checked and search for `[^\x00-\x7F]`



Find the unicode character and delete it for the `.Rmd` document to compile

Troubleshooting: Figures don't show

- My blog site on netlify doesn't show any of my figures

In the `R` console, run

```
blogdown::build_site()
```

Then open up a terminal window and check the git status

```
git status
```

There should be some un-tracked directories that end in the folder `/figure-html/`. Add these to git using

```
git add path/to/figure-html
```

and commit with

```
git commit -m "commit message"
```

Troubleshooting: Netlify builds but doesn't update blog

- My blog site on netlify builds without failure but doesn't show the updated blog posts

In the `R` console, run

```
blogdown::build_site()
```

Then open up a terminal window and check the git status

```
git status
```

There should be some un-tracked directories that end in the folder `/figure-html/`. Add these to git using

```
git add path/to/figure-html
```

and commit with

```
git commit -m "commit message"
```

Troubleshooting: Netlify build fails

To Do: add screenshot of build error message

- If you add data for analysis (`.csv` files), you need to tell `blogdown` to ignore the `.csv` files. To do this, open the `config.toml` file and modify the line:

```
ignoreFiles = ["\\.Rmd$", "\\.Rmarkdown$", "_cache$", "\\.knit\\.md$", "\\.utf8\\.md$"]
```

by adding `"\\.csv$"` to tell `blogdown` to ignore `.csv` files

so that the `ignoreFiles` line is now

```
ignoreFiles = ["\\.Rmd$", "\\.Rmarkdown$", "_cache$", "\\.knit\\.md$", "\\.utf8\\.md$", "\\.csv$"]
```

Troubleshooting: Netlify build fails

- If you add data for analysis (`.csv` files), the Netfliy build might fail, pointing to a `.csv` file

```
2:44:23 PM: Building sites … Total in 151 ms
2:44:23 PM: Error: Error building site: "/opt/build/repo/data/tidytuesday/CODE_OF_CONDUCT.md:1:1": unmarshal of format "" is not
supported
2:44:23 PM:
2:44:23 PM: ──────────────────────────────────────────────────────────────
2:44:23 PM:   "build.command" failed
2:44:23 PM: ──────────────────────────────────────────────────────────────
2:44:23 PM:
2:44:23 PM:   Error message
2:44:23 PM:   Command failed with exit code 255: hugo
```

To fix this, you need to tell `blogdown` to ignore the `.csv` files. To do this, open the `config.toml` file and modify the line:

```
ignoreFiles = ["\\.Rmd$", "\\.Rmarkdown$", "_cache$", "\\.knit\\.md$", "\\.utf8\\.md$"]
```

by adding `"\\.csv$"` to tell `blogdown` to ignore `.csv` files

so that the `ignoreFiles` line is now

```
ignoreFiles = ["\\.Rmd$", "\\.Rmarkdown$", "_cache$", "\\.knit\\.md$", "\\.utf8\\.md$", "\\.csv$"]
```

Troubleshooting: Netlify tidytuesday error

- If you moved data from tidytuesday into your blog repository, the netlify build might fail with the following error message.

```
2:44:23 PM: Building sites … Total in 151 ms
2:44:23 PM: Error: Error building site: "/opt/build/repo/data/tidytuesday/CODE_OF_CONDUCT.md:1:1": unmarshal of format "" is not
supported
2:44:23 PM:
2:44:23 PM: ──────────────────────────────────────────────────────────────
2:44:23 PM:   "build.command" failed
2:44:23 PM: ──────────────────────────────────────────────────────────────
2:44:23 PM:
2:44:23 PM:   Error message
2:44:23 PM:   Command failed with exit code 255: hugo
```

- If you added the tidytuesday repository analysis you need to tell `blogdown` to ignore the `.csv` files. To do this, open the `config.toml` file and modify the line:

```
ignoreFiles = [..., "\\.csv$"]
```

where the `...` is all the current values in this list . Then add `"tidytuesday"` to tell `blogdown` to ignore all files in the tidytuesday and recursive directories so that the `ignoreFiles` line is now

```
ignoreFiles = [..., "\\.csv$", "tidytuesday"]
```

Troubleshooting: Netlify gitmodules error

- The following error referencing gitmodules is caused by there being multiple `/.git/` respositories in your root blog directory (typically caused by pulling data from tidytuesday)

```
2:38:42 PM: Preparing Git Reference refs/heads/main
2:38:44 PM: Error checking out submodules: fatal: No url found for submodule path 'data/tidytuesday' in .gitmodules
2:38:44 PM: Failing build: Failed to prepare repo
2:38:44 PM: Failed during stage 'preparing repo': Error checking out submodules: fatal: No url found for submodule path
'data/tidytuesday' in .gitmodules
: exit status 128
2:38:44 PM: Finished processing build request in 4.979518168s
```

- First, identify where the offending `/.git/` directory is by navigating to the root project directory in the terminal then typing

```
find . -type d -name '.git'
```

which should return something like

```
./git
./data/tidytuesday/.git
```

Troubleshooting: Netlify gitmodules error

We have to remove the offending `/.git/` directory. Do this in the terminal by typing (making sure to modify the path to the correct path returned by the `find` command -- `./data/tidytuesday/` in the example)

```
rm -rf ./path/to/.git
```

Then we remove the directory containing the `/.git/` directory we just removed from tracking by git using (notice the two `--`s before cached and the path returned by `find`)

```
git rm -rf --cached ./path/to/
```

Type

```
git status
```

to confirm that the correct path has been deleted from tracking by git. Then add the deleted directory back into tracking by git with

```
git add ./path/to/
```

Then commit and push. Verify that the website builds correctly on Netlify