Creating a ‘story website’ on GitHub

For some stories the BBC Shared Data Unit creates a website to make it easier for partners to explore the data and the related material (by creating searchable tables or pages dedicated to their area, for example). This document provides a guide to that process, including how to access and update the website.

The steps:

1. Create an R notebook which can be ‘knitted’ into a HTML page or pages (this should be called index.Rmd so that it knits to a page called index.html)
2. If you need a different page to present data for each area, create a ‘template’ R notebook and use parameterisation to generate HTML outputs from that for each area which are all linked from an index.html homepage
3. If you want to track usage, set up a Google Analytics account for the site and add the code to the HTML file(s).
4. Create a new GitHub account using a different email - choose a password that you don’t use on other accounts and share it with other members of the project team
5. Create a new GitHub repo on that account for the project
6. Create a new *branch* in that repo for the website material
7. Upload the webpages and other documents (e.g. images) to that branch
8. Use the settings on the GitHub repo to publish the branch as a GitHub Pages website
9. Copy the link and share!

Below are details about each part…

Create an R notebook which can be ‘knitted’ into a HTML page or pages

There are two types of website you might create for a project:

- A single-page website, such as this one for a story on NHS dentists (separate GitHub repo here)
- A multi-page website with a page for each area, such as this one for police conduct reports (the R files are in the GitHub repo here)

For either you will need an **index page** - the homepage for the website. To create this you will need to create a notebook in R (to create one go to *File* > *New File* > *R Notebook*). This should be called index.Rmd because when a HTML file is generated from the notebook (‘knitted’) it will take the same name - i.e. it will eventually generate a page called index.html

The page needs to be called index.html because this is what a browser looks for when it goes to a website address. The URL https://sduiopc.github.io/test1/ for example actually loads https://sduiopc.github.io/test1/index.html.

So if you don't call it index then the browser won't be able to find the page when it goes to the website address, and it will give a 404 error.

Once you've created the index.Rmd file, fill it with all the information you want to go on the index page, formatting that information into headings and subheadings, bold, italic, blockquotes, images, links and lists using RMarkdown.

You can add this code to ensure that code blocks are not included in the HTML version:

~~~
```{r set display to false, include=FALSE}
#this saves us having to include these parameters in every code chunk
knitr::opts_chunk$set(echo = F, warning = F, message = F)
```
~~~

And you can create interactive (searchable) JavaScript tables using R's datatables package.

Once the notebook is ready to be published as a HTML page, use the Knit button at the top of the notebook to *'Knit to HTML'* and the index.html version should appear in the same folder.

More on the process here.

Optional: Create a template R notebook for 'region' pages (parameterisation)

If you want the website to have multiple pages for different regions, authorities, etc. then you can do this using a process called **parameterisation**.

This involves creating a page for one of the regions/authorities but using a variable instead of the region/authority name throughout. A script is then created to loop through a list of regions/authorities and use each one in place of that variable, 'knitting' a HTML page each time. The end result is a series of pages where different values are inserted into sentences or used to generate tables.

You can find out more about parameterisation here.

Parameterisation can create very large HTML files, which means RStudio takes a long time to generate those (if you have one for every local authority, this can mean a whole day). This is because by default, each HTML page will contain all of the code it needs to work (including all the JavaScript libraries). That is useful when sending HTML pages, but for publishing online it will be quicker and more efficient to have that code stored in a single file that all the HTML files link to.

To do this, ensure that the settings in the template notebook include *self_contained: false* in the *output:* parameters, and a folder specified for the libraries, like so:

```
---
title: |

![](https://raw.githubusercontent.com/sduiopc/test1/branch1/L
NP%20Header-01.jpg){align=right width=40%}
  Local authority analysis: `r params$la`
output:
  html_document:
      df_print: paged
      self_contained: false
      lib_dir: site/libs
      #see
https://stackoverflow.com/questions/41237664/reduce-file-size
-of-r-markdown-html-output
params:
  la: "Barnet"
---
```

When the HTML files are generated, a folder will also be created (with the name that you specify) to store all the JavaScript libraries that they depend on.

Create a new GitHub account using a different email

To host the website, create a new GitHub account - this ensures that users of the BBC GitHub account cannot find it before the embargo.

A useful technique is to take your Gmail address and insert a period in it and use that to create the account, e.g. p.aulonhismobile@gmail.com is treated by GitHub as a different email to paulonhismobile@gmail.com (but emails sent to both addresses will still reach me).

Consider choosing a username that refers to the subject. For example, for the NHS dentists story we simply chose ‘nhsdentists’. This then becomes the first part of the URL, i.e. nhsdentists.github.io

Create a new GitHub repo on that account for the project

Once the account is created, create a new repository (repo) for this project. You might simply use the name ‘getthedata’ as this will be used for the final part of the website address, i.e. nhsdentists.github.io/getthedata

Make sure you tick the box to ‘create a README file’.

Create a new *branch* in that repo for the website material

Once you've created the repo, create a new branch for the repo by clicking on the dropdown menu just above the list of files, marked ***'master'*** (this is the master branch).

This will open up a box where you can type the name of a new branch to create it.

Once you've typed the name you should see underneath it ***'Create branch [NAME] from 'master'***'

Click on that message to create the new branch. This will copy the repo into the new branch, and you will now be in that branch (you can tell which branch you are in by looking at the dropdown menu with the list of branches).

Upload the webpages and other documents (e.g. images) to that branch

Now, while in that branch, upload any HTML files by clicking and dragging them onto the page, and then following the process for uploading files to the repo.

Use the settings on the GitHub repo to publish the branch as a GitHub Pages website

Once all the files are in the repo, publish the website by going into the repo's Settings menu (click on the settings button in the upper right, or just add /settings to the URL) and scrolling down to the GitHub Pages section (or just add /settings/pages to the URL).

On that page under the 'Source' section change the dropdown menu from 'None' to the branch you created for the website.

Click **Save**.

A message should appear at the top of this section saying ***Your site is ready to be published at https://_______.github.io/____/*** - the account name will be in the first part of the URL and the repo name will be in the second part of the URL.

Click on the URL to test it. It may take a minute or two for the HTML files to register, so don't worry if it doesn't work immediately.

Make a copy of that URL and share it.

Optional: Add Google Analytics tracking code

If you want to track how much the website is used, create a Google Analytics tracking code by going to https://analytics.google.com/ and then the Admin section to *'Create property*'. (This process is detailed here). You'll need to give it a name and specify that you're measuring web traffic, then give the URL.

Once you've been through the process you'll be given some code to insert into your HTML page - it will look something like this:

```
<!-- Global site tag (gtag.js) - Google Analytics -->
<script async
src="https://www.googletagmanager.com/gtag/js?id=G-XNFWY0DN96"></script>
<script>
  window.dataLayer = window.dataLayer || [];
  function gtag(){dataLayer.push(arguments);}
  gtag('js', new Date());

  gtag('config', 'G-XNFWY0DN96');
</script>
```

To get this into your webpage, create a separate HTML file by selecting *File > New File > HTML File*.

Paste the Google Analytics code in the new file.

Save it with the name analyticsscript.html

Next, in your index.Rmd file, find the section at the start of the file with the 'meta data' about the document. Look for the part that says `output:  html_document:`

Underneath that, indented, add includes: `in_header: analyticsscript.html` so that the section as a whole looks like this (note the different level of indents)

```
output:
      html_document:
            includes:
                  in_header: analyticsscript.html
```

That last line specifies that the HTML file should be fetched and inserted inside the <head> tags.

You can read about this process at
https://stackoverflow.com/questions/41376989/how-to-include-google-analytics-in-an-rmarkdown-generated-github-page