Code

Accelerate web site repairs: An R Markdown script for use with spidering and traffic data

by Dan Wendling; last updated: 8/14/2017

Before using Run All: Update file names below for any file names that will/should change. After posting source files, Find read_csv and check file names here against the ones in your source file folder. Then Find write.csv and determine whether you want to change these file names.

Future work: Run a research project to determine what size image might be “too big” for smartphones, etc.: pageCountWithOverweightImg

MySQL: add noAltCountPg, noAltData

All opinions are my own.

Intro and start-up

Use this annotated RStudio Notebook script when you want to unite and summarize SEO Spider output and join it with site architecture, governance, and traffic information, to create actionable assignments for your web work team, including scoping work, prioritizing work, implementing, QA’ing, and posting site revisions. Annotations are meant to be a “lab notebook” with links to resources useful to modifying this script. This script can help you answer the questions, “What kind of trouble are we in?” (after a new web site problem or directive emerges), and “What content should we repair first?”

The output of this particular script allows for a new kind of web site quality analytics that is enterprise-readable, interactive, and free, by way of the D3.js tree map I have posted to GitHub, although Tableau, Power BI, Excel, etc. could also be used. The result will be two CSV files:

1. groupData - Information summarized at the page group / communication package level, by ownership and product/service. It summarizes broken links; the number of pages not recently reviewed; the number of images that may be too large for smart phones; etc. whatever your web site “problem of the moment” might be.
2. pageData - Information summarized at the page level: the specific broken link data, the specific dates of review, the specific images that are too large for smart phones, etc.

More about the script: This is a mostly “tidyverse” (and within that, mostly dplyr) script. Resources describing this include the online book and cheat sheets *R for Data Science*, by Hadley Wickham and Garrett Grolemund, http://r4ds.had.co.nz/ (http://r4ds.had.co.nz/) (also available in print from O’Reilly), and Garrett Grolemund’s O’Reilly videos *Expert Data Wrangling with R*, http://shop.oreilly.com/product/0636920035992.do (http://shop.oreilly.com/product/0636920035992.do).

I have been improving some procedure or other every use; this may not be the current version. Items that use “pd1” as input for “pd2” mean that I am avoiding over-writing things so I can view the old and new data frames, to make sure the appropriate changes were made. When troubleshooting is completed, I may start updating data frames instead of creating modified duplicates. All column names that begin with a capital letter are unaltered from the source; all column names that begin with a lower-case letter have been derived in R.

Information about the jsonlite package (used for export): https://cran.r-project.org/web/packages/jsonlite/index.html (https://cran.r-project.org/web/packages/jsonlite/index.html).

This file: This is an R Markdown (http://rmarkdown.rstudio.com) Notebook file. Create a new project in RStudio and put this Notebook in it. Run the commands one-by-one when you start, and modify the procedures to meet your data structure requirements and your business requirements. After you know what all procedures are doing and they meet your needs, you can use Run All when you’re in a hurry. When you execute code within the notebook, some results will appear beneath the code and some you will need to open the data frame to view. Shortcuts for editors:

- Run chunk: *Ctrl+Shift+Enter*
- Add chunk: *Ctrl+Alt+I*
- Add magrittr pipe %>% : *Ctrl + Shift + M*
- Learn about a command: ?select (as one example). And getting the dplyr cheat sheets is highly recommended.
- View data frame in Excel-like table: Click the item in the Environment pane within RStudio; see View > Panes.

When you save the notebook, an HTML file containing the code and output will be saved alongside it (click the *Preview* button or press *Ctrl+Shift+K* to preview the HTML file).

First-time users

1. In RStudio, select File > New Project. Create / select the folder you will use for files in this project. When you run the project through RStudio you will have convenient access to the Environment and Help panes (among others).
2. Add a chunk for install.packages(“tidyverse”, “stringr”, “lubridate”, “forcats”, “ggvis”, “jsonlite”). ggvis is optional and jsonlite has not been added here yet.

SEO Spider exports

Below are the exports I use from SEO Spider. More on exporting: https://www.screamingfrog.co.uk/seo-spider/user-guide/general/#exporting (https://www.screamingfrog.co.uk/seo-spider/user-guide/general/#exporting)

- ‘Internal’ tab > Filter: HTML > Export: internal_html.csv (required)
- If you’re NOT worried about running out of memory: Bulk Export > All Inlinks: all_inlinks.csv (for SWFs; using for now instead of all_flash.csv)
- Link diagnostics:
    - Bulk Export > Response Codes > Client Error (4xx) Inlinks: client_error_(4xx)_inlinks.csv

    - Bulk Export > Response Codes > Server Error (5xx) Inlinks: server_error_(5xx)_inlinks.csv
    - Bulk Export > Response Codes > Redirection (3xx) Inlinks: redirection_(3xx)_inlinks.csv
    - Bulk Export > Response Codes > Redirection (JavaScript) Inlinks: redirection_(javascript)_inlinks
    - Bulk Export > Response Codes > Redirection (Meta Refresh) Inlinks: redirection_(meta_refresh)*inlinks.csv New in SEO Spider 8, I have not investigated redirection*(javascript)*inlinks.csv and redirection*(meta_refresh)_inlinks.csv
- Bulk Export > All Images > all_images.csv
- Bulk Export > Images > Images Missing Alt Text Inlinks: images_missing_alt_text_inlinks.csv

Potentially useful; should be addressed in the future: - internal_pdf.csv - internal_other.csv - response_codes_all.csv - all_outlinks.csv - canonical_errors.csv

More info on exporting is below.

I am running five custom extractions in SEO Spider, such as email contact address and several dates managed in our web CMS. Contact me if you want more information about setting up custom extractions.

Using site traffic in decision making is tricky

Site traffic tells us important things, but can be tricky to extract meaning from, so it should be handled carefully. This script aggregates traffic up to the page group and organizational ownership levels, so decisions based on the single-page level can be avoided or consideration can be widened. In Google Analytics: Behavior > Site Content > All Pages; I update the dates to export 12 months of data. Future: My wish is to export 12 months and then derive an average per month, dividing by the number of months where data exists, on the “communication package” level rather than the individual page level. This ensures that seasonal swings are not left out and will provide a truer comparison of old content to brand new content. HOWEVER, for now I am exporting 12 months and dividing by 12, the most expedient calculation. The most useful number to me would be what the projected traffic will be in the next few months, based on past performance. Preparing for October and February when students are assigned tutorials, are different for us that other months. That scheme would have issues as well as what I“m doing now.

Then on the report page, Pages > Export > Unsampled Report. One of the options impacts the import code below, which is configured that the export includes “Metadata: Include report configuration details in header rows.” So several months down the line I can look at the exported files and not be confused about what the contents are.

Environment

If needed, clear all data frames from your work environment (Environment pane, see View > Panes in RStudio). Data frames require memory.

Hide

```
rm(list=ls())
```

If you don’t know your default working directory…

It’s hard to reliably import and export files if you haven’t set a default working directory. This happens automatically if you have established an RStudio project that you copy the inbound files into, but you might still want to verify it. If needed, set a new working directory through the menu options, or with a command such as setwd (“Q:/_/_webDS/R“). Windows, setwd(”C:/Documents/Text mining“); Mac, setwd (”/Users/wendlingd/Documents/webDS/R“)

Hide

```
getwd()
```

Start your session

Hide

```
library(tidyverse)
library(stringr)
library(lubridate)
library(forcats)
library(ggvis) # But you can comment this out and use ggplot2 if you want to, which will be load
ed with tidyverse. More on tradeoffs in the Visualization section.
# library(jsonlite)
```

==========

1. Create pageData table (data frame) from SEO Spider files

Bring in the page data.

'skip' means to ignore row 1 - because SEO Spider puts the name of the file on row 1.

Hide

```
internal_html <- read_csv("SEO_Spider/internal_html.csv", skip = 1)
```

Wrangle the page data

View in new tab as you would view an Excel worksheet:

Hide

```
# View(internal_html)
```

But it will probably be easier for you to click on the data frame names within the Environment pane. Use glimpse(item) as a commend if you want to see the data types column list for a data frame. Next:

1. Set column names to what they will be in MySQL.
2. Remove unneeded rows. The SEO Spider report includes URLs that failed to load during spidering (3xx and 4xx status codes). R explanation: http://r4ds.had.co.nz/transform.html#filter-rows-with-filter (http://r4ds.had.co.nz/transform.html#filter-rows-with-filter)
3. Select only the variables (columns) we need. More: https://www.r-bloggers.com/the-complete-catalog-of-argument-variations-of-select-in-dplyr/ (https://www.r-bloggers.com/the-complete-catalog-of-argument-variations-of-select-in-dplyr/)

Hide

```
pd1 <- internal_html %>%
  rename(`Title` = `Title 1`,
         `MetaDescription` = `Meta Description 1`,
         `MetaKeyword` = `Meta Keyword 1`,
         `DateIssued` = `DateIssued 1`,
         `DateModified` = `DateModified 1`,
         `DateExpiration` = `DateExpiration 1`,
         `ContactEmail` = `ContactEmail 1`,
         `DocumentType` = `DocumentType 1`,
         `RedirectUrl` = `Redirect URI`) %>%
  filter(`Status Code` == 200) %>%
  select(Address, Title, DateIssued, DateModified, DateExpiration, DocumentType, ContactEmail, R
edirectUrl, MetaDescription, MetaKeyword) %>%
  arrange(Address)
```

If you want info about the columns and structure,

Hide

```
glimpse(pd1)
```

Remove junk rows (If SEO Spider is reporting on non-www.mysite.gov URLs)

After running a project to compare URLs ending in slash, the folder name, to URLs ending in a variation of index-dot-something (index.html, index.htm, index.asp, index.cfm, etc.), I determined that the URLs ending in slash can be removed. To re-run this project see IsItSafeToRemoveURLsEndingInSlash-Yes.txt. Future: See if SEO Spider can stop duplicating these entries.

Hide

```
pd2 <- pd1 %>%
  filter(!grepl("/$", pd1$Address))
```

Hide

```
pd2 <- pd2 %>%
  filter(grepl("^https://www.mysite.gov", pd2$Address))
```

Remove URLs containing a ? or a # - Future: try to eliminate in SEO Spider

Hide

```
pd2 <- pd2 %>%
  filter(!grepl("\\?", pd2$Address))
```

Hide

```
pd2 <- pd2 %>%
  filter(!grepl("\\#", pd2$Address))
```

Build broken links data

Bring in

7/20/17: My machine has enough memory to process all_inlinks.csv, which allows me to consolidate several operations in this script, which I may switch to. Not all workstations have the memory to process all_inlinks.csv; for now I will take the easier route, although I have to bring in all_inlinks.csv for the PDF processing, later.

6/8/17: the 4xx file has 689 parsing failures. Can this be fixed?

Hide

```
LinkError4xx <- read_csv("SEO_Spider/client_error_(4xx)_inlinks.csv", skip = 1)
```

Hide

```
LinkError5xx <- read_csv("SEO_Spider/server_error_(5xx)_inlinks.csv", skip = 1)
```

Combine into one data frame

Hide

```
brokenLinks <- bind_rows(LinkError4xx, LinkError5xx, id = NULL)
```

Remove junk observations (junk rows)

Hide

```
ourBrokenLinks <- brokenLinks[grep("www.mysite.gov", brokenLinks$Source), ]
```

Hide

```
ourBrokenLinks <- filter(ourBrokenLinks, Anchor != "Redirect")
```

Hide

```
ourBrokenLinks <- filter(ourBrokenLinks, !grepl("/archive/", Source))
```

Update variable names (columns)

http://stackoverflow.com/questions/21752425/dplyr-mutate-in-r-add-column-as-concat-of-columns (http://stackoverflow.com/questions/21752425/dplyr-mutate-in-r-add-column-as-concat-of-columns)

Hide

```
  cleanOurBrokenLinks <- ourBrokenLinks %>%
  mutate(anchorDest = paste(Anchor, Destination, sep = ': ')) %>%
  select(Source, anchorDest, Destination, Anchor, `Status Code`)
```

Derive brokenLinkCountByPage

Hide

```
brokenLinkCountByPage <- cleanOurBrokenLinks %>%
  count(Source, sort = TRUE) %>%
  rename(bLinkCount = n)
```

Derive brokenLinkDataByPage

Verbose information about what the broken links are, aggregated up to the page level.
http://stackoverflow.com/questions/28752805/use-dplyr-to-concatenate-a-column (http://stackoverflow.com/questions/28752805/use-dplyr-to-concatenate-a-column)

Hide

```
brokenLinkDataByPage <- cleanOurBrokenLinks %>%
  group_by(Source) %>%
    summarise(bLinkData = paste(anchorDest, collapse = ". "))
```

Hide

```
pd3 <- pd2 %>%
  left_join(brokenLinkCountByPage, c("Address" = "Source"))
```

Add variable brokenLinkDataByPage

Hide

```
pd4 <- pd3 %>%
  left_join(brokenLinkDataByPage, c("Address" = "Source"))
```

If pd3 looks okay, that's all we need to retain in the environment. You can start over-writing things instead of creating new names, when you are sure no mistakes are getting in.

Hide

```
rm(brokenLinkCountByPage)
rm(brokenLinkDataByPage)
rm(brokenLinks)
rm(cleanOurBrokenLinks)
rm(internal_html)
rm(LinkError4xx)
rm(LinkError5xx)
rm(ourBrokenLinks)
rm(pd1)
rm(pd2)
rm(pd3)
```

Wrangle SWF data to create swfCountByPage

The file internal_flash.csv does NOT include the source URL (where the SWF was linked from), so for now I am getting SWF-related information from all_inlinks.csv.

Bring in

Huge file! 6/8/17: 112,400 parsing failures… How to resolve? but 8/4/17, 2.3M rows, imported okay (SEO Spider 8.1 recently installed)

Hide

```
InLinks <- read_csv("SEO_Spider/all_inlinks.csv", skip = 1)
```

Export table of http links for our http-to-https conversion

Hide

```
httpLinks <- InLinks[grep("^http://", InLinks$Destination), ] %>%
  arrange(Destination)
```

Hide

```
httpLinks2 <- httpLinks[grep("mysite.gov", httpLinks$Destination), ] %>%
  arrange(Destination)
```

Huge file (~ 1/2 million lines) - takes a long time to write

Hide

```
# write.csv(httpLinks2, file = "finished/httpLinks.csv", na="")
```

SWFs (Adobe Flash)

Hide

```
SWFInLinks <- filter(InLinks, grepl("swf$", Destination))
```

We are targeting SWF videos for retirement, however the exhibitions area uses Flash technology that can't be migrated to our media player as the "targeted" SWF videos can be; so files residing in /exhibition/ are left out for now.

Hide

```
SWFNotExhib <- filter(SWFInLinks, !grepl("gov/exhibition", Source))
```

swfCountByPage

Hide

```
SWFCountTbl <- SWFNotExhib %>%
  select(Source, Destination) %>%
  count(Source, sort = TRUE) %>%
  rename(swfCountByPage = n)
```

Hide

```
pd5 <- pd4 %>%
  left_join(SWFCountTbl, c("Address" = "Source"))
```

If pd5 is okay, clear the memory

Hide

```
rm(httpLinks)
#rm(httpLinks2)
rm(InLinks)
rm(pd4)
rm(SWFCountTbl)
rm(SWFInLinks)
rm(SWFNotExhib)
```

Wrangle image data

Hide

```
images <- read_csv("SEO_Spider/all_images.csv", skip = 1)
```

ImageCountPage

Hide

```
ImageCountTbl <- images %>%
  count(Source, sort = TRUE) %>%
  rename(imageCountPage = n)
```

Hide

```
pd6 <- pd5 %>%
  left_join(ImageCountTbl, c("Address" = "Source"))
```

totImageKiloPage

Create new column that converts Bytes to Kilobytes to reduce processing load and prevent “integer overflow” error

Hide

```
ImgSizeToKilo <- images %>%
  rename(`Bytes` = `Size (Bytes)`) %>%
  filter(!is.na(Bytes)) %>%
  mutate(ImageKiloPage = round(Bytes / 1000))
```

Get summary weight for the page

Hide

```
ImageSizeTbl <- ImgSizeToKilo %>%
  group_by(Source) %>%
  summarise(totImageKiloPage = sum(ImageKiloPage)) %>%
  arrange(desc(totImageKiloPage)) %>%
  ungroup()
```

Hide

```
pd7 <- pd6 %>%
  left_join(ImageSizeTbl, c("Address" = "Source"))
```

overweightImageCount - count of images per page that are over 200k

Count of images on each page that we consider to be too big for smartphones. SEO Spider has a report of images over 100k; this might be too low for our site type; setting to a 200k (200000 bytes) threshold for now…

Hide

```
ImgOverweightByPage <- ImgSizeToKilo %>%
  filter(!is.na(Bytes), Bytes > 200000 ) %>%
  count(Source, sort = TRUE) %>%
  rename(overweightImageCount = n)
```

Hide

```
pd8 <- pd7 %>%
  left_join(ImgOverweightByPage, c("Address" = "Source"))
```

Images lacking alt text, noAltCountPg, noAltData

Hide

```
noAltText <- read_csv("SEO_Spider/images_missing_alt_text_inlinks.csv", skip = 1)
```

8/9/2017: 13k rows list the Source *folder* rather than the source URL. I assume that SOME of these are dupes. Future: Project to determine whether to count these, or not. Removing for now.

Hide

```
noAltText2 <- noAltText %>%
  filter(!grepl("/$", noAltText$Source)) %>%
  arrange(Source)
```

Hide

```
noAltTextcountByPage <- noAltText2 %>%
  count(Source, sort = TRUE) %>%
  rename(noAltCountPg = n)
```

Hide

```
noAltDataByPage <- noAltText2 %>%
  group_by(Source) %>%
    summarise(noAltData = paste(Destination, collapse = ". "))
```

Hide

```
pd9 <- pd8 %>%
  left_join(noAltTextcountByPage, c("Address" = "Source"))
```

Hide

```
pd10 <- pd9 %>%
  left_join(noAltDataByPage, c("Address" = "Source"))
```

If pd10 looks okay,

Hide

```
rm(ImageCountTbl)
rm(images)
rm(ImageSizeTbl)
rm(ImgOverweightByPage)
rm(ImgSizeToKilo)
rm(pd5)
rm(pd6)
rm(pd7)
rm(pd8)
rm(pd9)
```

==========

2. Apply site architecture and governance structure / organizational ownership

If you put this data directly into your web CMS, you can scrape it from pages using SEO Spider's Custom Extract feature, and skip this section!

Now, assign to each row of the table (32,000+ pages in this case):

1. The name of the org chart piece that owns this information
2. The name of the microsite / communication package that the page belongs to (around 400 in this case)

This is what allows R to build a top-level summary for the web site. Easiest to build in a database, but Excel would also work. In my version, one row in this file will correspond to one row in internal_html.csv. I maintain this in a database, where it's easiest to update. If you don't have a database, Excel will be fine (but with extra work), and if you remember to save as CSV. My arrangement, row 1 and a sample row:

Address,contentSteward,stewardTopCat,stewardCode,contentGroup,templateName
"https://www.mysite.gov/index.html (https://www.mysite.gov/index.html)","OD-PA","PA",4,"Research Fellowship Program","MainTemplate"

Address: The URL of the page you are cataloging. Meant to be joinable with any other data regarding repair/diagnostic/traffic/etc.

contentSteward: A human-readable but short hierarchical path, starting from the top of the organization and listing organizational units “down” to the most specific team name that has permission to remove the content / is responsible for the content. In the example, OD-PA, means Office of the Director > Public Affairs. This tag is used in database queries and in JavaScript controls about ownership/governance, and in some chart labels. The name ‘Product Manager’ may also apply here.

stewardTopCat: Dictates where in the overall site summary chart this data will appear. The top category (org chart entry) that maintains the page in the web CMS. Content “owned” by an organizational division is not edited by the office of the director, so the division should be listed here. Whereas contentSteward locates the lowest team responsible, the stewardTopCat locates the highest group responsible.

stewardCode: A numeric representation of stewardTopCat for use by the database / JavaScript.

contentGroup: Name of the thematic group to which this URL belongs; the product or service that a group of pages is about. Other names could be the communication package, the microsite name, the content group, etc.

templateName: Some of our web work is to make pages responsive based on their web CMS template, change to an archive/retirement template, etc. Knowing the web CMS designation for the page type can be helpful.

Hide

```
IAGovStruct <- read_csv("IA_and_Governance_Struct/IA-Gov_2017-08-08.csv")
```

Just for your FYI, how the columns are indexed

Hide

```
glimpse(IAGovStruct)
```

Add IA/Gov in

Hide

```
pd11 <- pd10 %>%
  left_join(IAGovStruct, c("Address" = "Address"))
```

Check for entries with no assignment, repair and re-run as needed

Look at pages that lack the new assignments. If necessary, update the csv file and re-run this section.

Hide

```
pd11NoAssignment <- pd11 %>%
  filter(is.na(stewardTopCat)) %>%
  arrange(Address)
```

View the above in RStudio; best case: there are zero rows shown. Any entries here will NOT be tagged for IA and governance unless you update the mapping file and run this whole procedure again.

Hide

```
write.csv(pd11NoAssignment, file = "IA_and_Governance_Struct/updateTheseInMasterFile.csv",
na="")
```

If pd11 looks okay (check pd11NoAssignment!)

First round of mods complete for the data frame describing our pages. Change name to “pageData”

Hide

```
pageData <- pd11
```

Hide

```
rm(IAGovStruct)
rm(pd10)
rm(pd11)
rm(pd11NoAssignment)
```

==========

3. Bring in traffic data from Google Analytics

A connection to Google Analytics can be set up in SEO Spider, making this step unnecessary; see https://www.screamingfrog.co.uk/seo-spider/user-guide/configuration/#google-analytics-integration (https://www.screamingfrog.co.uk/seo-spider/user-guide/configuration/#google-analytics-integration). For now I’m using GA’s standard feature of exporting an unsampled report from GA, and then importing it here.

Future objective: Derive a 12-month average number of Unique Page Views per month. Because some of our content is seasonal, I think this will elevate important content to attention, that might otherwise be missed.

Use the ‘skip = 6’ argument when you told GA at export that you wanted the strategy included in the file.

8/4/17: Testing with 10 months of data. Will switch to 12 when our account goes back that far.

Hide

```
trafficNew <- read_csv("GAP/Pages_2017-08.csv", skip = 6)
```

Check to see if GAP reports folder names rather than the complete address

SEO Spider data includes folderName/fileName. These won’t be join-able if there are GAP entries with merely folderName/.

Standardize variable (column) names, add aveMonthlyUniquePV

Below assumes a 10-month export from Google Analytics! Change the ‘divided by’ as needed.

Hide

```
cleanTraff1 <- trafficNew %>%
  rename(`Address` = `Page`,
         `TotalPageViews` = `Pageviews`,
         `UniquePageViews` = `Unique Pageviews`,
         `AveTimeOnPage` = `Avg. Time on Page`,
         `BounceRate` = `Bounce Rate`,
         `PercentExit` = `% Exit`,
         `PageValue` = `Page Value`) %>%
  mutate(aveMonthlyUniquePV = round(UniquePageViews / 10)) %>%
  select(Address, aveMonthlyUniquePV, UniquePageViews, TotalPageViews, Entrances, BounceRate, Pe
rcentExit)
```

Require Address to start with www.mysite.gov/

Lots of entries for Google Translate, etc., that are interesting but I'm not studying translation now. Also lots of non-site entries - not sure where those are coming from.

Hide

```
cleanTraff2 <- cleanTraff1[grep("^www.mysite.gov/", cleanTraff1$Address), ]
```

Example of removing unneeded entries

Eventually I would like to use GAP to get accurate counts, and currently there is a lot of junk in the file, including malformed URLs and deleted/moved pages (if we do long timespans this could become a processing issue).

Hide

```
cleanTraff3 <- filter(cleanTraff2, !grepl("www.mysite.gov/unneededFolder", Address))
```

Create report of folder names in GAP

Hide

```
# GapUrlEndsWithFolderName <- cleanTraff3 %>%
#  filter(grepl("/$", cleanTraff3$Address))
```

Hide

```
#write.csv(GapUrlEndsWithFolderName, file = "finished/GapUrlEndsWithFolderName.csv", na="")
```

Make traffic Address join-able

Add https:// (https://) to start of Address

https://blog.exploratory.io/7-most-practically-useful-operations-when-wrangling-with-text-data-in-r-7654bd9d1a0c (https://blog.exploratory.io/7-most-practically-useful-operations-when-wrangling-with-text-data-in-r-7654bd9d1a0c). Uses library(stringr), loaded at top.

Hide

```
cleanTraff4 <- cleanTraff3 %>%
  mutate(`Address` = str_replace(`Address`, "^www", "https://www"))
```

Add index.html to end of Address when the GA URL ends with the directory, i.e. /

Hide

```
cleanTraff5 <- cleanTraff4 %>%
  mutate(`Address` = str_replace(`Address`, "/$", "/index.html"))
```

Add traffic to the pageData data frame

http://r4ds.had.co.nz/relational-data.html#mutating-joins (http://r4ds.had.co.nz/relational-data.html#mutating-joins)

Hide

```
pageData <- pageData %>%
  left_join(cleanTraff5, by = "Address")
```

Add page rank

Note: Pages with the same traffic count will have the same rank count. This is intentional; it's a "tie" when pages have the same traffic.

http://r4ds.had.co.nz/transform.html#grouped-summaries-with-summarise (http://r4ds.had.co.nz/transform.html#grouped-summaries-with-summarise)

Hide

```
pageData <- pageData %>%
  mutate(trafficRankPage = min_rank(desc(UniquePageViews)))
```

If pageData is okay, you can free up memory

Hide

```
rm(cleanTraff1)
rm(cleanTraff2)
rm(cleanTraff3)
rm(cleanTraff4)
rm(cleanTraff5)
rm(trafficNew)
```

==========

4. Create groupData - a summary data frame

New data frame: groupData with basics

contentGroup, contentSteward, pageCountGr (Total static HTML pages published)

Hide

```
gd1 <- pageData %>%
  group_by(contentGroup, contentSteward, stewardTopCat, stewardCode) %>%
  summarise(n = n()) %>%
  rename(`pageCountGr` = n) %>%
  select(contentGroup, contentSteward, stewardCode, stewardTopCat, pageCountGr)
```

FUTURE: Need to find a way to (1) view unmapped pages; (2) update mappings to collect unmatched rows; (3) integrate the new ones, or re-rerun from the start.

docTypesUsed

Results in, “Adding missing grouping variables.” Seems to run correctly, though.

Hide

```
docTypesUsed <- pageData %>%
  select(contentGroup, contentSteward, DocumentType) %>%
  group_by(contentSteward, contentGroup) %>%
  distinct(DocumentType) %>% select(DocumentType) %>%
  summarise(docTypesUsed = paste(DocumentType, collapse = " | "))
```

Hide

```
gd2 <- gd1 %>%
  left_join(docTypesUsed, by = c("contentGroup" = "contentGroup", "contentSteward" = "contentSteward"))
```

emailsUsed

Hide

```
emailsUsed <- pageData %>%
  select(contentGroup, contentSteward, ContactEmail) %>%
  group_by(contentSteward, contentGroup) %>%
  distinct(ContactEmail) %>% select(ContactEmail) %>%
  summarise(emailsUsed = paste(ContactEmail, collapse = " | "))
```

Hide

```
gd3 <- gd2 %>%
  left_join(emailsUsed, by = c("contentGroup" = "contentGroup", "contentSteward" = "contentSteward"))
```

aveMonthlyUniquePVsGr and trafficRankGr

Hide

```
UniquePVsGr <- pageData %>%
  filter(!is.na(UniquePageViews)) %>%
  group_by(contentGroup, contentSteward) %>%
  summarise(aveMonthlyUniquePVsGr = sum(UniquePageViews)) %>%
  arrange(desc(aveMonthlyUniquePVsGr)) %>%
  ungroup() %>%
  mutate(trafficRankGr = min_rank(desc(aveMonthlyUniquePVsGr)))
```

Hide

```
gd4 <- gd3 %>%
  left_join(UniquePVsGr, by = c("contentGroup" = "contentGroup", "contentSteward" = "contentStew
ard"))
```

bLinkCntGr

Hide

```
brokenLinkTotsGr <- pageData %>%
  filter(!is.na(bLinkCount)) %>%
  group_by(contentGroup, contentSteward) %>%
  summarise(bLinkCntGr = sum(bLinkCount)) %>%
  arrange(desc(bLinkCntGr)) %>%
  ungroup()
```

Hide

```
gd5 <- gd4 %>%
  left_join(brokenLinkTotsGr, by = c("contentGroup" = "contentGroup", "contentSteward" = "conten
tSteward"))
```

swfCountGr

Hide

```
SWFCountTots <- pageData %>%
  filter(!is.na(swfCountByPage)) %>%
  group_by(contentGroup, contentSteward) %>%
  summarise(swfCountGr = sum(swfCountByPage)) %>%
  arrange(desc(swfCountGr)) %>%
  ungroup()
```

Hide

```
gd6 <- gd5 %>%
  left_join(SWFCountTots, by = c("contentGroup" = "contentGroup", "contentSteward" = "contentSte
ward"))
```

imageCountGr

Hide

```
imageTots <- pageData %>%
  filter(!is.na(imageCountPage)) %>%
  group_by(contentGroup, contentSteward) %>%
  summarise(imageCountGr = sum(imageCountPage)) %>%
  arrange(desc(imageCountGr)) %>%
  ungroup()
```

Hide

```
gd7 <- gd6 %>%
  left_join(imageTots, by = c("contentGroup" = "contentGroup", "contentSteward" = "contentStewar
d"))
```

imageWeightMBGr

Hide

```
imageWeightTots <- pageData %>%
  filter(!is.na(totImageKiloPage)) %>%
  group_by(contentGroup, contentSteward) %>%
  summarise(imageWeightMBGr = sum(totImageKiloPage) / 1000) %>%
  arrange(desc(imageWeightMBGr)) %>%
  ungroup()
```

Hide

```
gd8 <- gd7 %>%
  left_join(imageWeightTots, by = c("contentGroup" = "contentGroup", "contentSteward" = "content
Steward"))
```

pageCountWithOverweightImg from overweightImageCount

For now we're defining overweight as more than 200k (200,000 bytes, see overweightImageCount above). SEO Spider has a report for images over 100k.

Hide

```
imageTotsOverweight <- pageData %>%
  filter(!is.na(overweightImageCount)) %>%
  group_by(contentGroup, contentSteward) %>%
  summarise(pageCountWithOverweightImg = sum(overweightImageCount)) %>%
  arrange(desc(pageCountWithOverweightImg)) %>%
  ungroup()
```

Hide

```
gd9 <- gd8 %>%
  left_join(imageTotsOverweight, by = c("contentGroup" = "contentGroup", "contentSteward" = "con
tentSteward"))
```

expiredPagesCntGr (Expired pages)

Hide

```
nowExpired <- pageData %>%
  filter(!is.na(DateExpiration), !is.na(contentSteward), DateExpiration <  ymd(20170601)) %>%
  group_by(contentGroup, contentSteward) %>%
  summarise(expiredPagesCntGr = n()) %>%
  arrange(desc(expiredPagesCntGr)) %>%
  ungroup()
```

Hide

```
gd10 <- gd9 %>%
  left_join(nowExpired, by = c("contentGroup" = "contentGroup", "contentSteward" = "contentStewa
rd"))
```

If gd10 looks okay, remove,

Hide

```
rm(brokenLinkTotsGr)
rm(docTypesUsed)
rm(emailsUsed)
rm(gd1)
rm(gd2)
rm(gd3)
rm(gd4)
rm(gd5)
rm(gd6)
rm(gd7)
rm(gd8)
rm(gd9)
rm(imageTots)
rm(imageTotsOverweight)
rm(imageWeightTots)
rm(nowExpired)
rm(SWFCountTots)
rm(UniquePVsGr)
```

recentEditsCntGr (Pages edited recently - Last 3 months, from DateModified)

Date calc for this and next uses today's date and the lubridate package. https://blog.exploratory.io/filter-with-date-function-ce8e84be680 (https://blog.exploratory.io/filter-with-date-function-ce8e84be680). You could also hard-code a date to go back to, such as "DateModified > ymd(20170308)".

Hide

```
recentEdits <- pageData %>%
  filter(!is.na(DateModified), !is.na(contentSteward), DateModified >= today() - days(90)) %>%
  group_by(contentGroup, contentSteward) %>%
  summarise(recentEditsCntGr = n()) %>%
  arrange(desc(recentEditsCntGr)) %>%
  ungroup()
```

Hide

```
gd11 <- gd10 %>%
  left_join(recentEdits, by = c("contentGroup" = "contentGroup", "contentSteward" = "contentStew
ard"))
```

recentAddsCntGr

Hide

```
recentAdds <- pageData %>%
  filter(!is.na(DateIssued), !is.na(contentSteward), DateIssued >= today() - days(90)) %>%
  group_by(contentGroup, contentSteward) %>%
  summarise(recentAddsCntGr = n()) %>%
  arrange(desc(recentAddsCntGr)) %>%
  ungroup()
```

Hide

```
gd12 <- gd11 %>%
  left_join(recentAdds, by = c("contentGroup" = "contentGroup", "contentSteward" = "contentStewa
rd"))
```

repairTotCnt

Future: Add a total repair count? But SQL can do this: UPDATE content_group SET repairTotCount = brokenLinksGr + recordingCountSWFGr + nonHtmlCountGr;

If gd12 is okay,

Hide

```
groupData <- gd12
```

Hide

```
rm(gd10)
rm(gd11)
rm(gd12)
rm(recentAdds)
rm(recentEdits)
```

Put the columns/variables into a logical reading order

UPDATE THIS EVERY TIME YOU ADD A NEW COLUMN. Use this when you want to create a more-logical column order for people using the data in R, Excel, etc. If you add more columns in the future, you can run glimpse to review the column order.

Hide

```
pageData = pageData %>%
  select(Address, Title, stewardTopCat, stewardCode, contentSteward, contentGroup, ContactEmail,
 aveMonthlyUniquePV, trafficRankPage, DateIssued, DateModified, DateExpiration, bLinkCount, bLin
kData, swfCountByPage, imageCountPage, totImageKiloPage, noAltCountPg, noAltData, overweightImag
eCount, DocumentType, RedirectUrl, MetaDescription, MetaKeyword, UniquePageViews,
TotalPageViews, Entrances, BounceRate, PercentExit)
```

Hide

```
groupData = groupData %>%
  select(contentGroup, pageCountGr, stewardCode, stewardTopCat, contentSteward, aveMonthlyUnique
PVsGr, trafficRankGr, bLinkCntGr, swfCountGr, imageCountGr, imageWeightMBGr, pageCountWithOverwe
ightImg, expiredPagesCntGr, recentEditsCntGr, recentAddsCntGr, docTypesUsed, emailsUsed)
```

==========

5. A few visualizations

Resources for viz in R

ggplot2

- Well-established with many features, including making static, complex, publishable graphs; mature. Can be output to png image.
- http://ggplot2.tidyverse.org/reference/ (http://ggplot2.tidyverse.org/reference/); site is http://ggplot2.org/ (http://ggplot2.org/)
- http://flowingdata.com/tag/ggplot2/ (http://flowingdata.com/tag/ggplot2/)
- https://www.datacamp.com/courses/data-visualization-with-ggplot2-1 (https://www.datacamp.com/courses/data-visualization-with-ggplot2-1)

ggviz

- Newest (announced 6/2014); forward-looking tool for creating web-based, interactive visualizations, but not as mature as ggplot2. Can be used for static graphs as well. Can be output to HTML/CSS-compatible format

(SVG).

- http://ggvis.rstudio.com/ (http://ggvis.rstudio.com/) Global site nav not the greatest so:
- http://ggvis.rstudio.com/cookbook.html (http://ggvis.rstudio.com/cookbook.html)
- http://ggvis.rstudio.com/ggvis-basics.html (http://ggvis.rstudio.com/ggvis-basics.html)
- http://ggvis.rstudio.com/0.1/quick-examples.html (http://ggvis.rstudio.com/0.1/quick-examples.html)
- https://blog.rstudio.org/2014/06/23/introducing-ggvis/ (https://blog.rstudio.org/2014/06/23/introducing-ggvis/)
- Adding titles through a hack: https://stackoverflow.com/questions/25018598/add-a-plot-title-to-ggvis (https://stackoverflow.com/questions/25018598/add-a-plot-title-to-ggvis)

qplot

- "Quick plot," which makes "plot" from ggplot2, is a bit easier to use. qplot provides simple syntax for basic views of what's in the data set. Mature, but too limited for complex displays.
- http://ggplot2.tidyverse.org/reference/qplot.html (http://ggplot2.tidyverse.org/reference/qplot.html)
- To date the current work has not used qplot, and adding a new sytax would seem confusing. May help sometime, if you find code on the web/books/etc…

Comparisons, etc.

- https://stats.stackexchange.com/questions/117078/for-plotting-with-r-should-i-learn-ggplot2-or-ggvis (https://stats.stackexchange.com/questions/117078/for-plotting-with-r-should-i-learn-ggplot2-or-ggvis)
- http://flowingdata.com/2016/03/22/comparing-ggplot2-and-r-base-graphics/ (http://flowingdata.com/2016/03/22/comparing-ggplot2-and-r-base-graphics/)
- https://stackoverflow.com/questions/5322836/choosing-between-qplot-and-ggplot-in-ggplot2 (https://stackoverflow.com/questions/5322836/choosing-between-qplot-and-ggplot-in-ggplot2)

Examples:

- https://flowingdata.com/tag/r/ (https://flowingdata.com/tag/r/)
- http://www.r-graph-gallery.com/portfolio/ggplot2-package/ (http://www.r-graph-gallery.com/portfolio/ggplot2-package/)

vs D3.js: https://flowingdata.com/2016/10/07/learning-r-versus-d3-js-for-visualization/ (https://flowingdata.com/2016/10/07/learning-r-versus-d3-js-for-visualization/)

Bar charts:

http://ggplot2.tidyverse.org/reference/geom_bar.html (http://ggplot2.tidyverse.org/reference/geom_bar.html)
https://rpubs.com/escott8908/RGC_Ch3_Gar_Graphs (https://rpubs.com/escott8908/RGC_Ch3_Gar_Graphs)

Time series: https://www.r-bloggers.com/plotting-time-series-data-using-ggplot2/ (https://www.r-bloggers.com/plotting-time-series-data-using-ggplot2/)

Note about the date calculation used to create some charts

Several charts describe recent activities, using date calculations such as "today() - days(90)." These are most accurate right after a new data set has been created; if you're running these at the beginning of each reporting cycle, this will work. If not, you might consider different coding.

Charts: What's in these tables?

Page count by content steward

Hide

```
chOwnerPageCount <- groupData %>%
  group_by(stewardTopCat) %>%
  summarise(totPagesOwner = sum(pageCountGr)) %>%
  arrange(desc(totPagesOwner)) %>%
  select(stewardTopCat, totPagesOwner)
```

Hide

```
ggplot(chOwnerPageCount, aes(x = reorder(stewardTopCat, totPagesOwner), y = totPagesOwner)) + ge
om_bar(stat = "identity") + coord_flip() + labs(title = "Page count by content steward")
```

Throughout this section, viewing ggvis charts is optional. If you chose to install ggvis:

Hide

```
  chOwnerPageCount %>% ggvis(x = ~stewardTopCat, y = ~totPagesOwner) %>% layer_bars()
```

Traffic by owner (Average monthly unique page views)

Hide

```
chOwnerAveMoPV <- groupData %>%
  filter(!is.na(aveMonthlyUniquePVsGr), !is.na(contentSteward)) %>%
  group_by(stewardTopCat) %>%
  summarise(totAveMonthlyUniquePgViews = sum(aveMonthlyUniquePVsGr)) %>%
  arrange(desc(totAveMonthlyUniquePgViews)) %>%
  select(stewardTopCat, totAveMonthlyUniquePgViews)
```

Hide

```
ggplot(chOwnerAveMoPV, aes(x = reorder(stewardTopCat, totAveMonthlyUniquePgViews), y = totAveMon
thlyUniquePgViews)) + geom_bar(stat = "identity") + coord_flip() + labs(title = "Traffic by owne
r", subtitle = "(Average monthly unique page views)")
```

Top 20 communication packages by traffic (Average monthly unique page views)

Hide

```
chGroupAveMoPV <- groupData %>%
  filter(!is.na(aveMonthlyUniquePVsGr), !is.na(contentGroup)) %>%
  group_by(contentGroup) %>%
  summarise(totAveMonthlyUniquePgViewsGr = sum(aveMonthlyUniquePVsGr)) %>%
  arrange(desc(totAveMonthlyUniquePgViewsGr)) %>%
  select(contentGroup, totAveMonthlyUniquePgViewsGr) %>%
  slice(1:20)
```

Hide

```
ggplot(chGroupAveMoPV, aes(x = reorder(contentGroup, totAveMonthlyUniquePgViewsGr), y = totAveMo
nthlyUniquePgViewsGr)) + geom_bar(stat = "identity") + coord_flip() + labs(title = "Top 20 commu
nication packages by traffic", subtitle = "(Average monthly unique page views)")
```

Top editing activity by specific owner, email owner (templated pages, past 3 months)

Change this to stewardTopCat (add this to the df)

Hide

```
chEditsBySteward <- pageData %>%
  filter(!is.na(DateModified), !is.na(contentSteward), DateModified >= today() - days(90), DateM
odified <= today()) %>% # Outliers can appear when an automated process updates many pages at on
ce, something I don't need to measure. These can be removed by adding to filter - contentSteward
 != "OD-PA"
  group_by(contentSteward) %>%
  summarise(Edits = n()) %>%
  arrange(desc(Edits)) %>%
  select(contentSteward, Edits)
```

Hide

```
ggplot(chEditsBySteward, aes(x = reorder(contentSteward, Edits), y = Edits)) + geom_bar(stat =
"identity") + coord_flip() + labs(title = "Top editing activity by specific owner", subtitle = "
(templated pages, past 3 months)")
```

For ggvis, similar to above, but time-series line chart where you can filter by org owner.

Hide

```
chEditsByStewardViz <- pageData %>%
  filter(!is.na(DateModified), !is.na(contentSteward), DateModified >= today() - days(90), DateM
odified <= today()) %>%
  group_by(DateModified, contentSteward) %>%
  summarise(Edits = n()) %>%
  arrange(DateModified) %>%
  select(DateModified, contentSteward, Edits)
```

View all first with “filter” commented out. Then uncomment the filter and add EXACT name… Separators: use a comma when you want to use “and”; use a pipe | character when you want to use “or”

Hide

```
chEditsByStewardViz %>%
  #filter(contentSteward == "OD-LO-PSD-RWS-USU" | contentSteward == "OD-LO-PSD-RWS-WIM") %>%
  ggvis(~DateModified, ~Edits, stroke = ~contentSteward) %>% layer_points() %>%  layer_lines()
```

Edits by email address:

Hide

```
chEditsByEmailViz <- pageData %>%
  filter(!is.na(DateModified), !is.na(ContactEmail), DateModified >= today() - days(90), DateMod
ified <= today()) %>%
  group_by(DateModified, ContactEmail) %>%
  summarise(Edits = n()) %>%
  arrange(DateModified) %>%
  select(DateModified, ContactEmail, Edits)
```

Hide

```
chEditsByEmailViz %>%
  # filter(contentSteward == "OD-LO-PSD-RWS-USU" | contentSteward == "OD-LO-PSD-RWS-WIM") %>%
  # filter(ContactEmail != "wwwteam@mysite.gov") %>%
  # ggvis(~DateModified, ~Edits, stroke = ~ContactEmail) %>% layer_points() %>%  layer_lines()
```

Top editing activity by communication package (templated pages, past 3 months)

Hide

```
chEditsByContentGr <- groupData %>%
  filter(!is.na(recentEditsCntGr), recentEditsCntGr > 5, !is.na(contentSteward)) %>% # Place to
 remove outliers, see above
  arrange(desc(recentEditsCntGr)) %>%
  select(contentGroup, recentEditsCntGr)
```

Hide

```
  ggplot(chEditsByContentGr, aes(x = reorder(contentGroup, recentEditsCntGr), y = recentEditsCnt
Gr)) + geom_bar(stat = "identity") + coord_flip() + labs(title = "Top editing activity by commun
ication package", subtitle = "(templated pages, past 3 months)")
```

For ggvis, similar to above, but time-series line chart where you can filter for the communication packages of one org owner.

Hide

```
chEditsByCommPkgViz <- pageData %>%
  filter(!is.na(DateModified), !is.na(contentGroup), DateModified >= today() - days(90), DateMod
ified <= today(), contentSteward != "OD-PA") %>%
  group_by(DateModified, contentSteward, contentGroup) %>%
  summarise(Edits = n()) %>%
  arrange(DateModified) %>%
  select(DateModified, contentSteward, contentGroup, Edits)
```

Filter requires EXACT names, which can be found above. Separators: For “and” use comma; for “or” use pipe |

Hide

```
chEditsByCommPkgViz %>%
  filter(contentSteward == "OD-PA" | contentSteward == "OD-PA") %>%
  ggvis(~DateModified, ~Edits, fill = ~contentGroup) %>% layer_points()
```

Editing activity by date (templated pages, past 6 months)

Needed to restrict DateModified to dates today or before, because of outliers saying they were updated 2018-07. Also removing an outlier.

Hide

```
chEditsByDate <- pageData %>%
  filter(!is.na(DateModified), !is.na(contentSteward), DateModified >= today() - days(180), Date
Modified <= today(), contentGroup != "PA-Other") %>%
  group_by(DateModified) %>%
  summarise(Edits = n()) %>%
  select(DateModified, Edits)
```

Hide

```
ggplot(chEditsByDate, aes(x = DateModified, y = Edits)) + geom_bar(stat = "identity") + labs(tit
le = "Editing activity by date", subtitle = "(templated pages, past 6 months)")
```

`

Broken links by email owner

Still working on this.

chEditsByEmailViz <- pageData %>% filter(!is.na(DateModified), !is.na(ContactEmail), DateModified >= today() - days(90), DateModified <= today()) %>% group_by(DateModified, ContactEmail) %>% summarise(Edits = n()) %>% arrange(DateModified) %>% select(DateModified, ContactEmail, Edits)

chbLinksByContentGr <- pageData %>% filter(!is.na(bLinkCount), !is.na(contentGroup)) %>% group_by(ContactEmail) %>% summarise(bLinks = n()) %>% arrange(desc(bLinks)) %>% select(ContactEmail, bLinks) %>% filter(row_number() <= 20, row_number() >= 2)

Hide

```
chbLinksByEmail <- pageData %>%
  filter(!is.na(DateModified), !is.na(ContactEmail), DateModified >= today() - days(90), DateMod
ified <= today()) %>%
  group_by(ContactEmail) %>%
  summarise(bLinks = n()) %>%
  arrange(desc(bLinks)) %>%
  select(ContactEmail, bLinks)
```

Hide

```
  ggplot(chbLinksByEmail, aes(x = reorder(ContactEmail, bLinks), y = bLinks)) + geom_bar(stat =
"identity") + coord_flip() + labs(title = "Broken links by email owner")
```

Report on the ‘broken links repair’ initiative

Still working

What proportion of the broken links reported 30 days ago have been fixed?

Stacked bar, numbers not percent, by owner, to show volume by owner and the recent activity to fix them (or the lack of recent activity to fix them). Future: third element - height of top bar is the total count of links owned (excluding topnav, bottomnav, etc.)

Last month's broken links report This month's broken links report

Could be link count reported, link count resolved; or,

The number of pages that were edited to remove the broken links?

Hide

```
rm(chbLinksByContentGr)
rm(chEditsByDate)
rm(chGroupAveMoPV)
rm(chOwnerAveMoPV)
rm(chOwnerPageCount)
rm(EditsByContentGr)
rm(EditsByDate)
rm(EditsBySteward)
rm(OwnerPageCount)
rm(scatEdits)
```

==========

6. Write out database load files

http://r4ds.had.co.nz/data-import.html#writing-to-a-file (http://r4ds.had.co.nz/data-import.html#writing-to-a-file)

Hide

```
write.csv(pageData, file = "finished/page-2017-08.csv", na="")
write.csv(groupData, file = "finished/group-2017-08.csv", na="")
```

Hide

```
?write.csv # for more options
```