USWDS Difference in Differences analysis

library(tidyverse)

## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5 v purrr 0.3.4  
## v tibble 3.1.6 v dplyr 1.0.8  
## v tidyr 1.2.0 v stringr 1.4.0  
## v readr 2.1.2 v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

# library(did)  
library(lubridate)

##   
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':  
##   
## date, intersect, setdiff, union

filepath <- "G://Shared drives//MSG Projects//3.0 Digital Government (LQ3)//2233 USWDS Digital Dashboard//03. Data Collection//"  
colorName <- "color\_contrasts.csv"  
htmlName <- "html\_attribute.csv"  
imageName <- "missing\_image.csv"  
wdsName <- "uswds\_data.csv"

## Introduction

The U.S. Web Design System (USWDS) is a set of web design standards intended to make websites more accessible. In a prior study, the Office of Customer Experience and Office of Evaluation Sciences established a positive correlation between using key elements of the USWDS and higher website accessibility scores. This analysis is an extension of that effort to infer a causal relationship between use of the USWDS and increased website accessibility.

## Data

The current analysis relies on data gathered for digitaldashboard.gov.

There are four main components of the data:

1. html: this element counts the number of html attribute issues on a webpage.
2. Image: this element indicates the number of images with missing text descriptions.
3. Color: the element indicates the total number of color contrast issues.
4. USWDS: this element indicates whether any of the USWDS components are flagged in a scan of the webpage. This takes on either a yes or no value, although there are also “null” values that could indicate errors in the scan.

# need to convert date to date type.   
# There is a gap in scans between Nov 19,  
# 2020 and March 6, 2022  
  
# about 1/3 of observations are coerced to NA, will need to   
# take a closer look at values before deciding that is okay  
# or if should be zero. Looks like all that are coerced to   
# N/A have value "NULL" but there are also 0s in the data  
color <- read.csv(paste0(filepath, colorName)) %>%  
 rename(ContrastIssues=Number.of.Color.Contrast.Issues) %>%  
 mutate(Date2=as\_date(Date, format="%d-%b-%y"),  
 # ContrastIssues=as.numeric(ContrastIssues),  
 weekYear = floor\_date(Date2, unit="week"),  
 monthYear = floor\_date(Date2, unit="month"),  
 Week=week(Date2),  
 Year=year(Date2)) %>%  
 distinct(Website, weekYear, ContrastIssues, .keep\_all=TRUE)  
  
# three lags: website to indicate the first obs in the group;   
# contrastIssues to understand if they are duplicates; days b/w   
# scans to see if they are close together  
colorNoDup <- color %>%  
 arrange(Website, Date2) %>%  
 mutate(lag = lag(Date2), daysBetween = Date2-lag(Date2),  
 sameWebsite= (Website==lag(Website)),  
 sameIssues = (ContrastIssues==lag(ContrastIssues))) %>%  
 filter(!(daysBetween <=6 & daysBetween >=0 & sameWebsite==1 & sameIssues==1))  
  
# getting number of scans that occur close to others  
colorDup5 <- color %>%  
 arrange(Website, Date2) %>%  
 mutate(lag = lag(Date2), daysBetween = Date2-lag(Date2),  
 sameWebsite= (Website==lag(Website)),  
 sameIssues = (ContrastIssues==lag(ContrastIssues))) %>%  
 filter(daysBetween <=5 & daysBetween >=0 & sameWebsite==1) %>%  
 nrow()  
  
colorDup6 <- color %>%  
 arrange(Website, Date2) %>%  
 mutate(lag = lag(Date2), daysBetween = Date2-lag(Date2),  
 sameWebsite= (Website==lag(Website)),  
 sameIssues = (ContrastIssues==lag(ContrastIssues))) %>%  
 filter(daysBetween <=6 & daysBetween >=0 & sameWebsite==1) %>%  
 nrow()  
  
colorDup7 <- color %>%  
 arrange(Website, Date2) %>%  
 mutate(lag = lag(Date2), daysBetween = Date2-lag(Date2),  
 sameWebsite= (Website==lag(Website)),  
 sameIssues = (ContrastIssues==lag(ContrastIssues))) %>%  
 filter(daysBetween <=7 & daysBetween >=0 & sameWebsite==1) %>%  
 nrow()  
  
colorNAs <- sum(color$ContrastIssues=="NULL")

For an initial scan of the data, there appear to be several issues with the data. In most cases, the issues fit within two general types.

First, the scans do not appear to be fully automated. While our conversations with the digitaldashboard team indicated that scans should take place approximately every two weeks, there are both instances in which there are scans very close together (e.g., run within two days of each other) and some periods without comprehensive scans. When looking at the scan for color contrast issues, there are 28 scans within 5 days of each other, 33 scans within 6 days of each other, and 764 scans within 7 days of each other. The large jump in scans at 7 days apart suggests that there is one legitimate scan done on exactly a one week interval. In nearly all cases, the number of issues are the same for both scans, so deleting one observation from the pair could be considered equivalent to de-duplicating by week. I de-duplicate any scans that occur within 6 or fewer days of each other

The larger issue is that very few webpages have any html, image, or color scan data between the end of 2020 (either November or December) and May of 2022. This means there is only approximately a 6 month period in which we have both data on accessibility and an indicator for websites using USWDS.

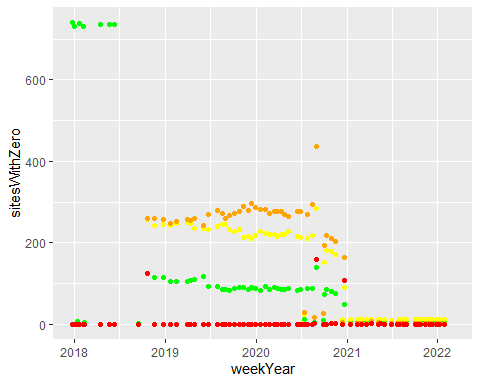
Second, there are a number of “Null” values included in the data. It is unclear what these values represent given that there also are zero values. It is possible for a website without an image to have a value that is not applicable, which perhaps indicates a “Null” response, but there are also high frequencies of null values for color contrast issues—10970— which do not seem to have the same likelihood for not applicable values. If the values are not applicable, and we are interested in catching problems when they do exist, we could coerce the “Null” values to zero. If the “Null” values indicate an error in the scan, it would be more appropriate to drop those observations. We treat the NA values as missing, and they are not used in any calculations.

html <- read.csv(paste0(filepath, htmlName))%>%  
 rename(Website=Website.Name, Date=date, HTMLIssues=Number.of.HTML.Aattribute.issues) %>%  
 mutate(Date2=as\_date(Date, format="%d-%b-%y"),  
 weekYear = floor\_date(Date2, unit="week"),  
 monthYear = floor\_date(Date2, unit="month")) %>%  
 # distinct(Website, Date2, Number.of.HTML.Aattribute.issues, .keep\_all=TRUE)  
 distinct(Website, weekYear, HTMLIssues, .keep\_all=TRUE)  
  
  
# View(htmlDups %>%  
# left\_join(html, by=c("Website", "Week", "Year")) %>%  
# arrange(Website, Year, Week))  
  
  
# There are a fair number of duplicate obs per week, but the same weeks  
# are popping up: 20/2018, 24/2018, 7/2020, some in 51/2020 and another couple  
# of random ones. Looks like June 13 and June 15, 2018 was two full scans,   
# feb 13 and feb 14 2020, dec 18 and dec 21, 2020 has a couple, but de-duping on   
# week/year and same outcome cuts dups down to 3  
  
  
# the exact duplicates are easy enough to delete  
  
image <- read.csv(paste0(filepath, imageName))%>%  
 rename(Website=Website.Name, ImageIssues=Number.of.Missing.Image.Issues) %>%  
 mutate(Date2=as\_date(Date, format="%d-%b-%y"),  
 weekYear = floor\_date(Date2, unit="week"),  
 monthYear = floor\_date(Date2, unit="month")) %>%  
 distinct(Website, weekYear, ImageIssues, .keep\_all=TRUE)  
  
# still a few duplicates in the week that have different values  
# one way to treat could be to take the mean or the min/max  
# imageDups <- getDups(image)  
# imageDups %>% group\_by(weekYear) %>% summarise(count = n())  
  
  
# wds scan starts in 25th week of 2020, so far fewer observations, only 52 weeks  
# also only 99 websites scanned instead of 1112 for others  
# maybe the correct order here is to use the 99 websites to join the accessibility   
# scores, but for wds set the value to 0 for any obs prior to when the scans started  
# although that may not be entirely accurate, or just have to restrict to those 52   
# weeks  
wds <- read.csv(paste0(filepath, wdsName))%>%  
 rename(Website=Website.Name) %>%  
 mutate(Date2=as\_date(Date, format="%d-%b-%y"),  
 weekYear = floor\_date(Date2, unit="week"),  
 monthYear = floor\_date(Date2, unit="month")) %>%  
 distinct(Website, weekYear, USWDS.Score, .keep\_all=TRUE)  
  
# looks like a problem in dec 2020 with a real? scan run on the 17/18 and something that   
# returned 22 out of 25 nulls on the 20.   
  
# there are some violations of the assumption once treated always treated. some sites have  
# either some scans that return 0 in a longer sequence of 100s and at least one goes from  
# a long string of 100s to 0s, which in theory is possible, but would I think be a violation of  
# the law  
  
# could find indicator for any USWDS.Score==100 and earliest date, merge that back and see if  
# there are any obs that are 0 or NULL after that Week, Year (logical)  
# wdsDups <- getDups(wds)  
# wdsDups %>% group\_by(weekYear) %>% summarise(count = n())  
# View(wdsDups %>%  
# left\_join(wds, by=c("Website", "weekYear")) %>%  
# arrange(Website, weekYear))  
  
# View(wds %>%   
# filter(Website %in% c("fedidcard.gov", "fsd.gov", "sftool.gov")) %>%  
# arrange(Website,Date2))  
  
# View(wdsDups %>%  
 # left\_join(wds, by=c("Website", "weekYear")) %>%  
 # group\_by(Date2, USWDS.Score) %>%  
 # summarize(Count=n()))  
  
# checking to see if there are any scans after the first appearance  
# of USWDS that then show a 0 value  
wdsMin <- wds %>%  
 filter(USWDS.Score==100) %>%  
 group\_by(Website) %>%  
 summarize(wdsMin = min(Date2))  
  
wdsSwitch <- wds %>%  
 left\_join(wdsMin, by="Website") %>%  
 mutate(revert = if\_else(Date2>wdsMin & USWDS.Score != 100, 1, 0))  
  
# View(wdsSwitch %>%   
# group\_by(Website, revert) %>%  
# summarize(count = n()) %>%  
# arrange(Website))  
#   
# look at those that have >=8 obs that USWDS.Score==0 after at least one ==100  
# think about forcing them to 100 after that date? or drop? maybe sensitivity   
# analysis for dropping first then including with forced 100 and caveating?  
# View(wds %>%  
# filter(Website %in% c("challenge.gov", "ecpic.gov", "fedidcard.gov",   
# "fleet.gov", "fpc.gov")) %>%  
# arrange(Website, Date2))  
  
# mostly a good match on the accessibility factors at the week level  
accessibility <- color %>%  
 left\_join(html, by=c("Website", "weekYear"), suffix=c("\_color","\_html")) %>%  
 left\_join(image, by=c("Website", "weekYear"), suffix=c("\_color","\_image")) %>%  
 left\_join(wds, by=c("Website", "weekYear"), suffix=c("\_color","\_wds"))   
  
# goo <- wds %>%  
# left\_join(html, by=c("Website", "Week", "Year")) %>%  
# left\_join(image, by=c("Website", "Week", "Year")) %>%  
# left\_join(color, by=c("Website", "Week", "Year"))  
  
# want to find out if the NULL values are grouped by scan or not  
# look at distribution of total NULL values by scan. Does not appear that it is   
# very likely that all three show up together, and fairly even between seeing   
# no nulls, one null, or two nulls  
  
  
nullTotals <- accessibility %>%  
 mutate(totNulls = (HTMLIssues=="NULL")+(ContrastIssues=="NULL")+(ImageIssues=="NULL"))  
  
table(nullTotals$totNulls, useNA='always')

##   
## 0 1 2 3 <NA>   
## 9245 9507 11250 569 11

nullTotalsTable <- nullTotals %>%  
 group\_by(weekYear) %>%  
 summarise(sitesWithZero = sum(if\_else(totNulls==0,1,0)),  
 sitesWithOne = sum(if\_else(totNulls==1,1,0)),  
 sitesWithTwo = sum(if\_else(totNulls==2,1,0)),  
 sitesWithThree = sum(if\_else(totNulls==3,1,0)))  
  
ggplot(nullTotalsTable, aes(x=weekYear, y=sitesWithZero)) +  
 geom\_point(aes(x=weekYear, y=sitesWithZero), color="green") +  
 geom\_point(aes(x=weekYear, y=sitesWithOne), color="yellow") +  
 geom\_point(aes(x=weekYear, y=sitesWithTwo), color="orange") +  
 geom\_point(aes(x=weekYear, y=sitesWithThree), color="red")

## Warning: Removed 6 rows containing missing values (geom\_point).  
## Removed 6 rows containing missing values (geom\_point).  
## Removed 6 rows containing missing values (geom\_point).  
## Removed 6 rows containing missing values (geom\_point).



getDups <- function(df) {  
 dups <- df %>%   
 group\_by(Website, weekYear) %>%  
 summarise(count=n()) %>%  
 filter(count>1)   
 return(dups)  
}  
  
htmlDups <- getDups(html)

## `summarise()` has grouped output by 'Website'. You can override using the  
## `.groups` argument.

htmlDups %>% group\_by(weekYear) %>% summarise(count = n())

## # A tibble: 1 x 2  
## weekYear count  
## <date> <int>  
## 1 2020-02-09 3

# View(htmlDups %>%  
# left\_join(html, by=c("Website", "Week", "Year")) %>%  
# arrange(Website, Year, Week))  
  
  
# There are a fair number of duplicate obs per week, but the same weeks  
# are popping up: 20/2018, 24/2018, 7/2020, some in 51/2020 and another couple  
# of random ones. Looks like June 13 and June 15, 2018 was two full scans,   
# feb 13 and feb 14 2020, dec 18 and dec 21, 2020 has a couple, but de-duping on   
# week/year and same outcome cuts dups down to 3  
# the exact duplicates are easy enough to delete  
  
  
imageDups <- getDups(image)

## `summarise()` has grouped output by 'Website'. You can override using the  
## `.groups` argument.

imageDups %>% group\_by(weekYear) %>% summarise(count = n())

## # A tibble: 1 x 2  
## weekYear count  
## <date> <int>  
## 1 2020-02-09 3

names(image)

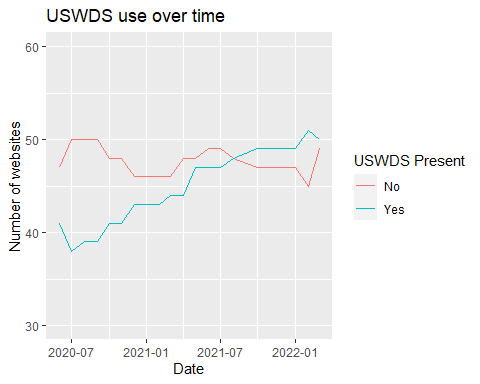
## [1] "Website" "ImageIssues" "Date" "Date2" "weekYear"   
## [6] "monthYear"

View(image %>%  
 mutate(missing=ImageIssues=="NULL")%>%  
 group\_by(year(monthYear)) %>%  
 summarise(missing=sum(missing)))  
  
imageNoNullByDate <- image %>%  
 filter(ImageIssues!="NULL") %>%  
 group\_by(weekYear) %>%  
 summarize(avgImageIssues=mean(as.numeric(ImageIssues))) %>%  
 right\_join(image, by=c("weekYear"))

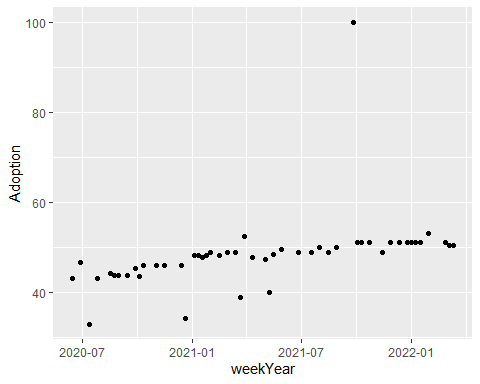
# show uswds uptake overtime (by month)  
  
wdsMonth <- wds %>%  
 filter(USWDS.Score!="NULL") %>%  
 group\_by(Website, monthYear) %>%  
 summarise(maxUSWDS = max(as.numeric(USWDS.Score))) %>%  
 ungroup() %>%  
 group\_by(monthYear, maxUSWDS) %>%  
 summarise(Count = n()) %>%  
 ungroup() %>%  
 filter(Count>1)

## `summarise()` has grouped output by 'Website'. You can override using the  
## `.groups` argument.  
## `summarise()` has grouped output by 'monthYear'. You can override using the  
## `.groups` argument.

ggplot(wdsMonth, aes(x=monthYear, y=Count,   
 color=as.factor(maxUSWDS))) +  
 geom\_line() +  
 ylim(30,60) +  
 labs(title="USWDS use over time", x="Date",   
 y="Number of websites", color = "USWDS Present") +  
 scale\_color\_discrete(labels = c("No", "Yes"))



# ggsave("uswdsSitesOverTime.png", device = "png")  
  
  
# show percent uswds adoption over time  
adoptionByWeek <- wds %>%   
 filter(USWDS.Score!="NULL") %>%  
 group\_by(weekYear) %>%  
 summarise(Adoption=mean(as.numeric(USWDS.Score)))  
  
ggplot(adoptionByWeek, aes(x=weekYear, y=Adoption)) +  
 geom\_point()

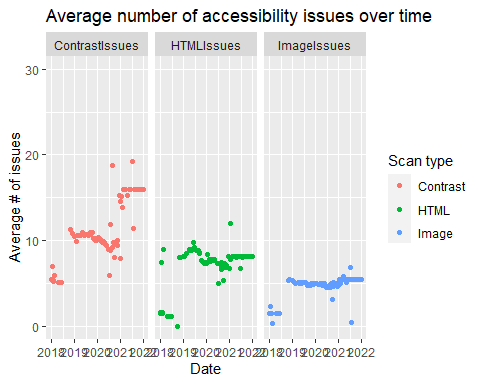


# facet avg number of accessibility issues over time  
# will need some transformation to put score type as column  
accessibilityLong <- accessibility %>%  
 pivot\_longer(cols=contains("ssues"), names\_to = "ScanType", values\_to = "Issues") %>%  
 filter(Issues!="NULL") %>%  
 group\_by(weekYear, ScanType) %>%  
 summarise(MeanIssues=mean(as.numeric(Issues), na.rm=TRUE))

## `summarise()` has grouped output by 'weekYear'. You can override using the  
## `.groups` argument.

issuePlot<-ggplot(accessibilityLong %>% filter(weekYear<"2022-01-01"),  
 aes(x=weekYear, y=MeanIssues, color=ScanType)) +  
 geom\_jitter() +  
 ylim(0,30) +  
 labs(title="Average number of accessibility issues over time",  
 x="Date", y="Average # of issues", color="Scan type") +  
 scale\_color\_discrete(labels=c("Contrast", "HTML", "Image"))  
  
issuePlot<-ggplot(accessibilityLong %>% filter(weekYear<"2022-01-01"),  
 aes(x=weekYear, y=MeanIssues, color=ScanType)) +  
 facet\_wrap("ScanType") +  
 geom\_jitter() +  
 ylim(0,30) +  
 labs(title="Average number of accessibility issues over time",  
 x="Date", y="Average # of issues", color="Scan type") +  
 scale\_color\_discrete(labels=c("Contrast", "HTML", "Image"))  
  
  
issuePlot

## Warning: Removed 2 rows containing missing values (geom\_point).



# ggsave("accessibilityIssuesOverTime.png", device = "png")

## Methods

## Results

## Threats to validity

## Recommendations