

# Sinclair Broadcasting and Racial Resentment: Evidence from Google Trends and Project Implicit

April 20, 2021

## Acknowledgments

## Abstract

# Contents

<b>1</b>	<b>Introduction</b>	<b>6</b>
<b>2</b>	<b>Theoretical Framework</b>	<b>7</b>
<b>3</b>	<b>Background</b>	<b>7</b>
3.1	Sinclair Broadcast Group . . . . .	7
3.2	What Characterizes Racially Conservative Coverage? . . . . .	8
3.2.1	Does Sinclair Have a Racial Bias? . . . . .	8
3.3	Measuring Racial Animus . . . . .	11
3.3.1	Google Trends Data . . . . .	11
3.3.2	Scaling Google Trends Data: . . . . .	12
3.4	Implicit Association Test data . . . . .	14
<b>4</b>	<b>Methodology</b>	<b>16</b>
4.1	Tools Used . . . . .	16
4.2	Preregistration . . . . .	17
4.3	Difference-In-Differences Analysis . . . . .	17
<b>5</b>	<b>Results</b>	<b>18</b>
5.1	Identification Assumption . . . . .	19
<b>A</b>	<b>Codings for Offensive Words</b>	<b>21</b>
<b>B</b>	<b>References</b>	<b>21</b>
<b>C</b>	<b>Preregistration Document</b>	<b>24</b>
C.1	Pregistration . . . . .	25
C.2	Pregistration . . . . .	26
C.3	Pregistration . . . . .	27
C.4	Pregistration . . . . .	28
<b>D</b>	<b>Code</b>	<b>29</b>
D.1	Master Replication Script . . . . .	29
D.2	Cleaning Sinclair Data . . . . .	29

D.3	Web Scraping . . . . .	30
D.4	Cleaning Google Search Data . . . . .	31
D.5	Cleaning IAT Data . . . . .	31
D.6	Running Models . . . . .	32
D.7	Code in This Document . . . . .	33
D.8	Utility Functions - R . . . . .	35
D.9	Utility Functions - Python . . . . .	35
D.9.1	Get Between-Region Google Search Data . . . . .	35
D.9.2	Get Between-Times Google Search Data . . . . .	36
D.9.3	Scaling Google Search Data . . . . .	37
D.10	Unit Tests . . . . .	39
D.10.1	For Python Code . . . . .	39
D.10.2	For R Code . . . . .	42

*This paper uses Google search terms containing offensive language as a proxy measure for racial animus. Following the practice of similar works (Stephens-Davidowitz, 2014; Chae, Clouston, Hatzenbuehler, et al., 2015; Chae, Clouston, Martz, et al., 2018; Isoya and Yamada, 2021), I use coded language to refer to these terms. The words themselves can be found in [Appendix A](#)*

## 1 Introduction

Can media coverage influence racial resentment? Empirical studies suggest that when White Americans understand welfare policies to threaten their privileged status in the U.S. Social hierarchy, their resentment of minorities increases and their support for welfare decreases (Willer, Feinberg, and Wetts, 2016; Wetts and Willer, 2018). This tendency seems to be weaponed by conservative media institutions and politicians, who encourage racial animus to erode support for social programs, as in the infamous example of the “Welfare Queen” narrative, a racial stereotype employed to undercut support for the Aid to Families with Dependent Children (AFDC). However, the link between traditional media coverage and racial resentment has not been extensively studied.

I use the expansion of the Sinclair Broadcasting Group from 2004-2021 to understand how conservative media messaging impacts racial resentment in a media market. The expansion of the Sinclair Broadcasting Group during this period provides the basis for a difference-in-differences analysis estimating the effect of Sinclair Media purchasing a station on racial animus in an area.

I use two internet-based measures of racial animus, the number of google searches containing racial epithets, and average scores on the IAT as measures of racial animus in an area. Google trends data have previously been used to measure the contributions of racial resentment towards African-American mortality (Chae, Clouston, Martz, et al., 2018; Chae, Clouston, Hatzenbuehler, et al., 2015), election outcomes (Stephens-Davidowitz, 2014), and economic inequality (Connor et al., 2019), and does not suffer from the same social censoring issues that confound traditional measures.

Overall, I find no link between Sinclair moving into a county and changes of racial resentment. This result is robust to multiple measurement strategies and specifications.

## 2 Theoretical Framework

Data on racial animus is scarce, and rife with social desirability and reporting biases. To circumvent these problems, I turn to two internet-based measures of racial animus: the number of Google searches for racial epithets in a region, and scores on the Harvard IAT.

I contribute to an existing body of research (DellaVigna and Kaplan, 2007; Miho, 2018; Gentzkow, 2006; Gentzkow, Shapiro, and Sinkinson, 2011; Morris, 2005) that uses the variation in station ownership caused by the expansion of major news institutions as the basis of a causal study.

Ladd and Lenz (2009) document four major hurdles to convincingly document several major hurdles to documenting media persuasion — a lack of major variation in media coverage, poor measures of media exposure, and alternative explanations for media effects (consumer self-selection and media outlet pandering towards consumer).

First, there is little variation in media messaging; news outlets have a financial incentive to carve out a segment of the market (Mullainathan and Shleifer, 2005), and so generally endorse candidates / employ framings sympathetic to a single party (Ladd and Lenz, 2009, p. 395; Ansolabehere, 2006, p. 13). Second, measures of media exposure are often primitive (Ladd and Lenz, 2009, p. 395).

Where effects are found, it can be difficult to understand whether they come as a result of media persuading consumers, consumers shifting their news consumption to align with their existing views, or media institutions shifting their coverage to better align with the interests of their readers (*ibid.*, p. 395).

DellaVigna and Kaplan (2007) and Ladd and Lenz (2009), use difference-in-differences approaches to estimate the effect of changes in media endorsement on election outcomes.

## 3 Background

### 3.1 Sinclair Broadcast Group

In this paper, I use the expansion of the Sinclair Broadcasting Group from 2004-2020 as the basis of a difference-in-differences analysis. Founded in 1971 as the Chesapeake Television Corporation, the Sinclair Broadcasting Group is one of the largest telecommunications providers in the U.S,

and has grown to serve the maximum 39% of households allowable under U.S. law (Scherer, 2018, p. 1).

Sinclair’s pattern of expansion stands out from other providers in the industry. Rather than purchasing stations in large media markets across the country, Sinclair has expanded by buying up stations in small media markets. Previous research has demonstrated that Sinclair acquisition of a network is associated with a sharp rightwards shift in its coverage (Martin and McCrain, 2019). I use the change in media coverage induced by Sinclair expansion as the basis of a difference-in-difference analysis of the contribution of media framing towards racial animus.

I create a record of Sinclair’s station ownership over time from the company’s yearly filings with the Securities and Exchange Commission, which enumerate all the stations owned by the company at the end of each year. This strategy has previously been used by Miho (2018) to track Sinclair Expansion over the period of 1995-2017. A map of the media markets Sinclair either purchased or sold a station in can be seen in in figure 1a. A chart showing when each station became treated can be seen in 1b.

Traditionally, researchers track a network’s expansion using datasets maintained by the Nielsen Corporation, the U.S. company responsible for tracking the boundaries of media markets. The SEC dataset has an advantage over this dataset as it is open-access, and can be published freely alongside the research.

## **3.2 What Characterizes Racially Conservative Coverage?**

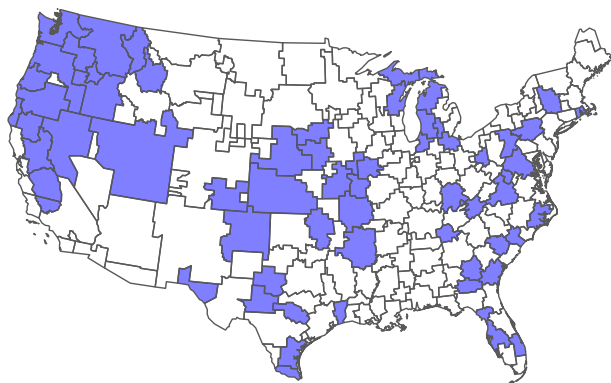
I use the term-racially conservative to denote a particular media framing characterized by its insistence that race does not shape modern-day Americans life prospects and “that individual characteristics, not structural barriers, explain group-based disparities” (Engelhardt’2019). In its most extreme forms, this framing instructs consumers that “demands from minority groups for special attention and improvements to their station” (Engelhardt’2019) are either without basis, attempts to defraud the welfare state, or calls to redress cultural failings of Black Americans.

### **3.2.1 Does Sinclair Have a Racial Bias?**

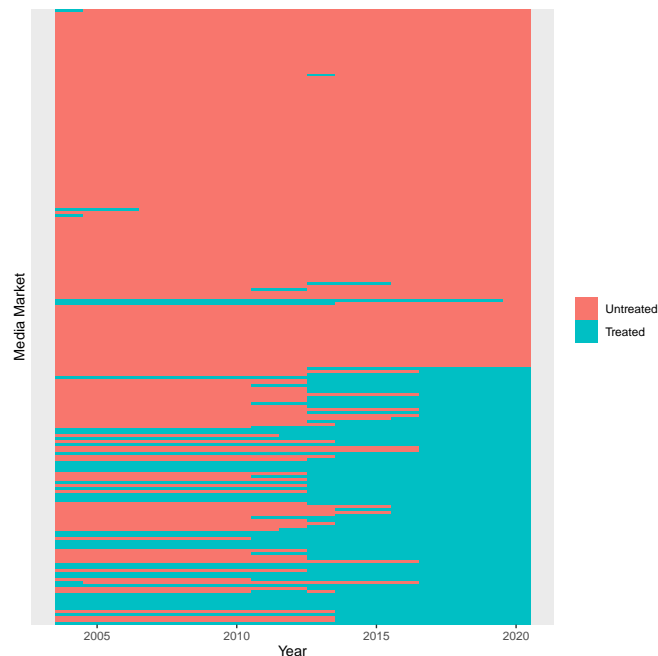
I assume that Sinclair stations differ from non-Sinclair stations on coverage of racial issues: if Sinclair stations deploy the same coverage on racial issues as other stations, then there is no causal pathway by which Sinclair acquisition can change levels of racial resentment in a market.



(a) Media Markets Sinclair Bought or Sold a Station in between 2004-2020. DMA Boundaries From Hill, 2015



(b) Media Markets by Sinclair Ownership Status, Time



In this section, I provide evidence to motivate this assumption. While Sinclair’s coverage of racial issues has not been extensively studied, research does show that Sinclair buying a station leads to a sharp rightwards shift in its coverage. I argue that this effect extends to the coverage of racial issues - when Sinclair buys a station, it shifts the coverage of racial issues sharply towards the right, and provide anecdotal evidence to support this assertion.

First, Sinclair explicitly models itself as a local-news alternative to Fox News, a station known for its racially-conservative coverage.

“Fox News Channel has demonstrated that people want a different level of truth, and if you can do it nationally, why not locally? If we’re successful in creating meaningful, relevant controversy, we’ll be doing a community service.”

Sinclair frequently pushes “must run” segments which are mandatory for owned stations to run. They often take the form of scripts local anchors are mandated to read, or pre-filmed segments that are broadcast over all Sinclair stations. Figure ?? shows images from one of these scripts being read on 30 stations and an accompanying transcript (ironically, a script about elite control of media institutions). These must-run segments often espouse racially conservative framings of news stories. I offer several examples of such must-run segments below.

Figure 2: Sinclair News Anchors Reading a “Must-Run” Script (May 2018)



(a) Images of 30 of 210 Sinclair Station Hosts Reading a Must-Run Script (Burke, 2018)

**(A):** But we’re concerned about the troubling trend of irresponsible, one sided news stories plaguing our country. The sharing of biased and false news has become all too common on social media.

**(B):** More alarming, some media outlets publish these same fake stories... stories that just aren’t true, without checking facts first.

**(A):** Unfortunately, some members of the media use their platforms to push their own personal bias and agenda to control ‘exactly what people think’...This is extremely dangerous to a democracy.

(b) Transcript of Segment (Cohen, 2018)

In 2010, Sinclair stations approved for broadcast and ran “Breaking Point,” an almost cartoonishly-racist, 25-minute campaign advertisement which, among other things, cast aspersions that Obama’s 2008 presidential campaign was financed by Hamas and implied Obama was a Muslim by juxtaposing clips of him saying “As-salamu alaykum” (peace be with you) in a speech in Cairo against audio of Islamic prayers (2:08-2:50,4:36-5:15 GOP Trust PAC, 2010).

This year, the Sinclair broadcasting has drawn ire for a series of “must-run” segments on police violence following the murder of George Floyd pushing the “Black-on-Black violence” canard and advocating for a military response to the protests (Pleat and Savillo, 2020; Pleat, 2020b). After Kyle Rittenhouse, a seventeen-year-old from Antioch, Illinois drove to Kenosha and allegedly murdered two protesters, Sinclair broadcast a segment to 93 stations warning that American cities were “on the edge of a race war not seen since the sixties,” and promoted the idea of neighborhood blockades and patrols to defend neighborhoods from threat (Pleat, 2020a)

These segments do not represent the entire spectrum of political programming broadcast over Sinclair stations, but the ease with which these segments can be found and the intense vitriol they contain point to a network that is extremely racially conservative.

### 3.3 Measuring Racial Animus

Measuring racial resentment is difficult. Overt expressions of racism are severely socially sanctioned in the U.S., so respondents who do harbor racial resentment often understate it when they are interviewed for traditional surveys (Krumpal, 2011; Kuklinski, Cobb, and Gilens, 1997). These social-desirability biases confound traditional survey measures, and lead to underestimates of the levels of racial hostility in the states.

I sidestep social desirability biases by using two unconventional measures of racial animus: the number of Google searches for racial slurs in an area, and test scores from Harvard University’s Project Implicit, an internet-based project to collect data on implicit biases. Both measures estimate racial animus from observed behavior (Google searches or the ability to associate Black and White faces with positive words) rather than asking respondents questions, so sidestep issues of self-reporting.

#### 3.3.1 Google Trends Data

I use the concentration of Google searches containing racial epithets as a proxy measurement of the level of racial animus in an area. Data from Google searches are ideal as searches are not subject to the social-desirability biases that confound traditional, survey-based measures of racial resentment, and provide a large, regularly-sampled source of data which represents a great deal of the population (at present, Google as an over 85% market share in the US).

The core assumption of this strategy is that Google searches for the word [Word 1] reflect underlying racial animus in an area. If readers find this assumption unconvincing, then they will have little reason to accept the conclusions of this analysis. Accordingly, I offer several reasons to suggest that searches for racial epithets well proxy racial animus in an area, namely: searches are conducted in private, and are not subject to the same social-desirability biases as traditional, survey-based measures; and measurements of area-level racism obtained from Google trends data correspond well to traditional survey-based and non-survey-based measures of racial animus.

First, Google searches do not suffer from the same social censoring as traditional measures of public opinion that rely on face-to-face or over-the-phone interviews. Overt expressions of racism are no longer socially palatable in the U.S., Social desirability bias is a pressing concern when it comes to measuring racial animus; research has repeatedly demonstrated that (Kuklinski, Cobb, and Gilens, 1997)

This measurement approach has an advantage over traditional survey-based measures of racial animus in that it is less subject to a social desirability bias; “Google searchers are online and likely alone, both of which make it easier to express socially taboo thoughts (Kreuter et al., 2009)” (Stephens-Davidowitz, 2014, p. 26). Further, it provides a high-resolution set of data that would be prohibitively expensive to collect from a traditional survey, especially given that the difference-in-differences approach requires a repeated survey comparable across multiple time periods.<sup>1</sup>

A pressing concern is that searches for the word “[Word 1]” might not actually measure racial animus but, in fact, reflect users learning about the term. Indeed, “Definition of [Word 1]” and “what does [Word 1] mean” are both among the top 5 search queries related to term. These queries suggest that many who search for the term are searching out of curiosity to investigate the term.

This is not to suggest that searches for the term do not capture any variation in racial animus: the top 10 most searched related queries to the term [Word 1]’s are “I don’t like [Word 1]’s,” “fuck the [Word 1]’s,” and “ship those [Word 1]’s back.” Stephens-Davidowitz (*ibid.*) evidences this claim by reporting strong correlations between opposition to interracial marriage and Google searches for [Word 1] at the state level. I provide further support for this claim by reporting correlations between searches for [Word 1] and the scores of White respondents in a state on the IAT. Following practices of other research using IAT data (Connor et al., 2019, p. 206) I use the scores of only White respondents.

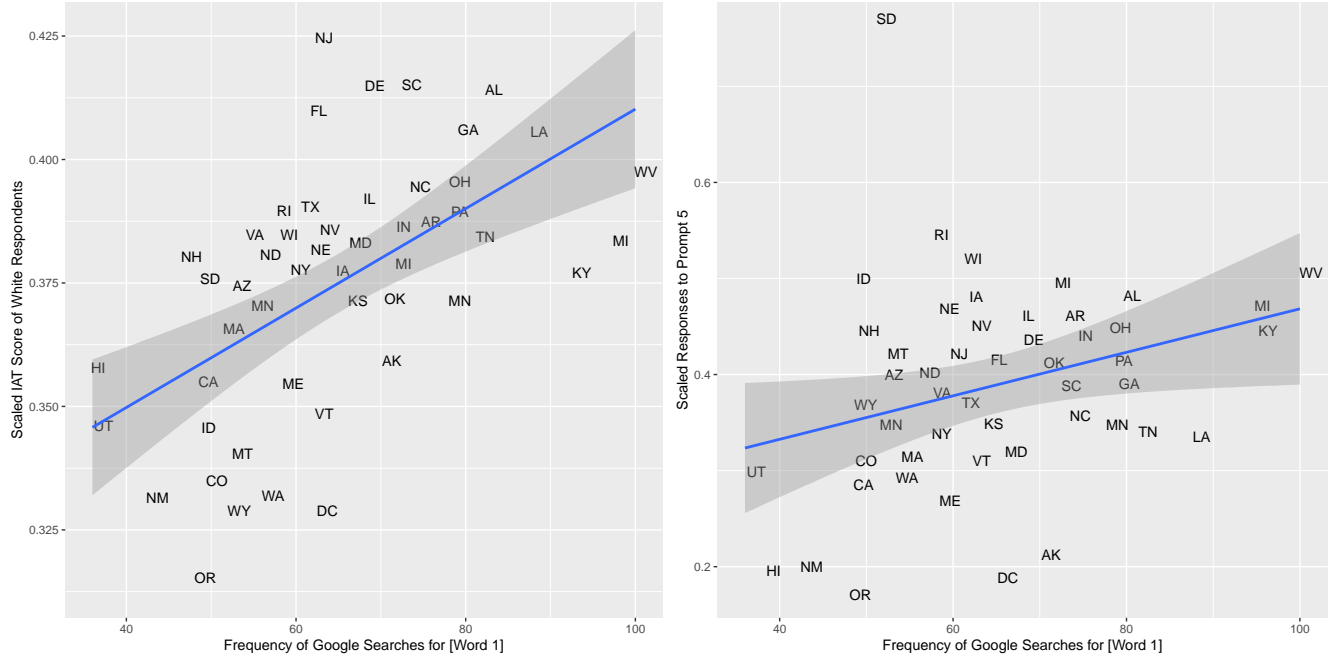
### 3.3.2 Scaling Google Trends Data:

Google trends data measures the popularity of a search on an idiosyncratic scale: a term’s “Google search score” in a given time is given as the volume of searches over that time divided by the volume of searches when the term was most popular. However, the difference-in-difference analysis I perform requires that the search volume is measured on the same scale. In this section, I describe the scaling process used to back out an interval-level measure of search popularity from these search scores.

Search popularity can be obtained from Google in two flavors: a measure that compares the

---

<sup>1</sup>Stephens-Davidowitz, 2014 evidences this claim by reporting statistics for pornography searches. Over the past 16 years, the number of searches for “porn” and “news” are commensurate, yet only 14% of GSS respondents tell the GSS they have visited a pornographic website in the past 30 days.



popularity of a search across all regions at a given time, and a measure that gives the popularity of a search in a given region across all times.

The search score that can be used to make comparisons between regions at a given time is defined as the following:

$$\text{Between Regions Search Score} = \frac{\text{Popularity in Region } i}{\text{Popularity in the Most Popular Region}}$$

The search score that can be used to compare the popularity across different times in a given region is defined as the following.

$$\text{Between Times Search Score} = \frac{\text{Popularity at Time } i}{\text{Popularity at the Most Popular Time}}$$

The first measure allows for comparisons between regions but not between times, while the second permits comparisons between times but only within one region. When combined together, these measures can be used to compare the volume of searches across both time periods and regions.

As an example, we might want to compare searches for the word apple in Washington D.C. in 2017 to searches in Pensacola in 2018. Washington D.C. had twice the number of searches for

the word apple as Pensacola in 2017. As Pensacola had three times the amount of searches for the word apple in 2017 as 2018, we know that Pensacola had just  $1/6th$  the searches for the word apple in 2018 as Washington D.C. in the year 2017.

By comparing the volume of searches in each region at each time to each other region, I back out a measure of search volume,  $v$  that is comparable both between regions and between times.

$$v_{rt} \propto \sum_{i=1}^I \sum_{j=1}^J \frac{v_{rt}}{v_{ij}}$$

For every  $v_{rt} \neq v_{i'j'}$ , where:

- $v_{ij}$  Represents the search volume in region  $i$  at time  $j$
- Regions  $j \in (1 \dots J)$
- Time periods  $i \in (1 \dots I)$

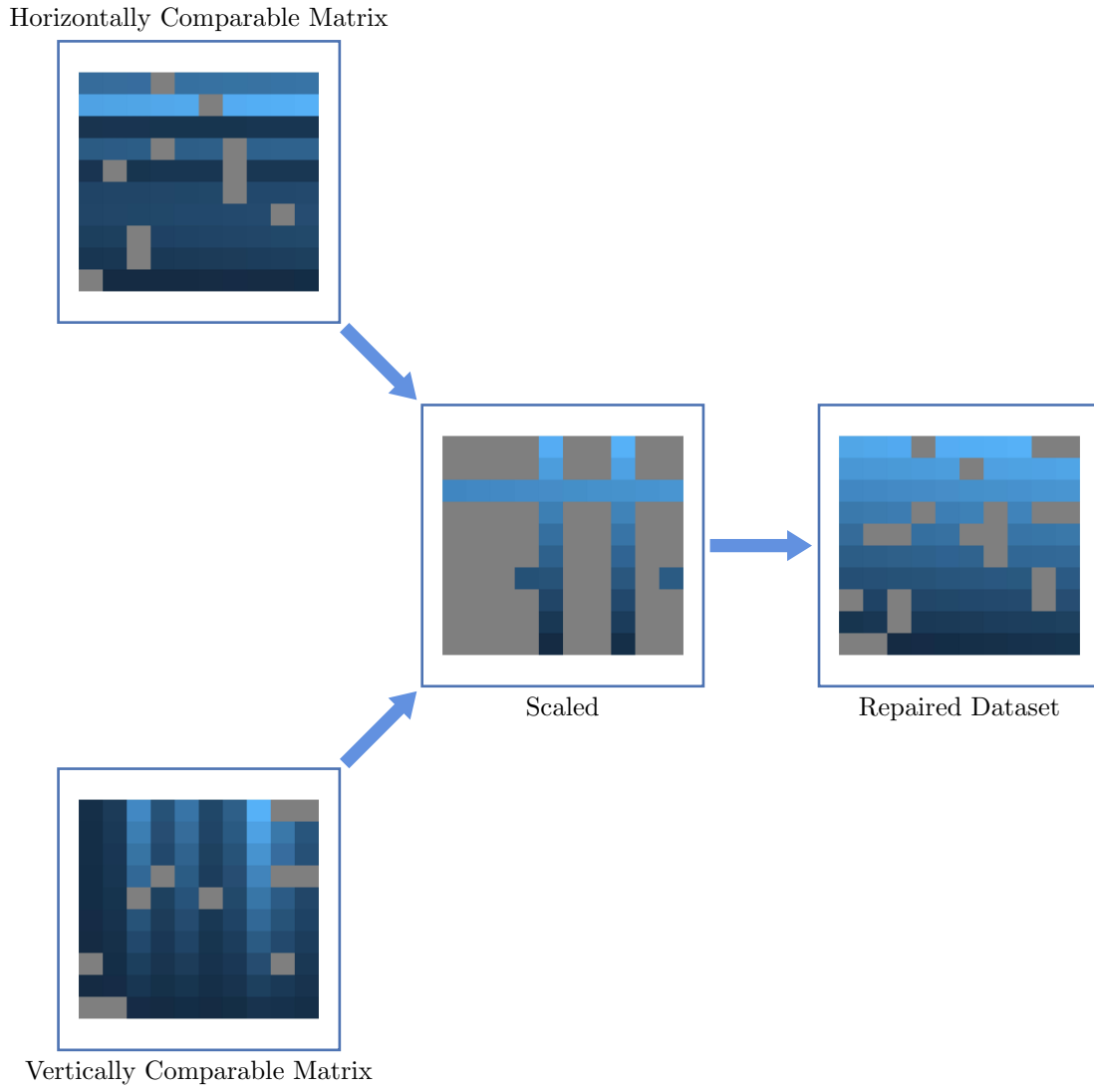
### 3.4 Implicit Association Test data

To provide a second measure of racial animus, I use data from the Harvard race IAT (Xu et al., 2021). The Harvard IAT is an internet-based quiz founded in 1998 that seeks to measure implicit associations between “concepts (e.g., Black people, gay people) and evaluations (e.g., good, bad)” (Xu et al., 2020).

The race IAT asks participants are sort good and bad words, and African-American and European-American into two categories. Examples of these faces and words can be seen in figure 5. The IAT records the difference in the speed with which respondents can sort European-American faces with good words / African-American faces with bad words, and European-American faces with bad words / African-American faces with good words.

The IAT records the differences in the speed with which respondents can sort European-American + Good / African-American + Bad together and European-American + Bad / African-American + Good together. The IAT would record a positive score (an implicit preference for European-Americans) if a respondent was quicker to code European-American + Good / African-American + Bad together than they were to code European-American + Bad / African-American + Good together. The underlying logic behind using the differences in speeds is that “making a

Figure 4: Illustration of Scaling Algorithm




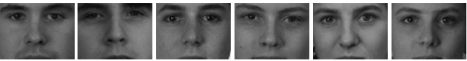
Category	Items
Good	Delightful, Excellent, Cheer, Happy, Appealing, Adore, Friend, Glad
Bad	Ugly, Disgust, Angry, Selfish, Humiliate, Detest, Nasty, Annoy
African Americans	
European Americans	

Figure 5: Slides From an IAT test (Xu et al., 2021)

response is easier when closely related items share the same response key [category]” (Xu et al., 2021), so respondents who have a strong implicit association between African-American and Bad will find it more difficult to code European-American and Good together.

The IAT responses are not subject to social-desirability biases as they are determined by implicit associations that respondents cannot control. In fact, methodological studies have shown that even when asked to create a certain results, participants are unable to devise or enact a strategy to get their desired results unless given specific instructions on how to do so by a researcher (Kim, 2003, pp. 88–91).

IAT scores also

Data from the IAT is recorded at the county level, I measure racial animus at the larger DMA level. To solve this issue, I aggregate IAT scores to the DMA level using a crosswalk of US counties to DMA ID’s from 2016 (Sood, 2016). In actual fact, DMA boundaries undulate slightly over time to include and exclude new counties, and a lack of publicly-accessible historical data means I cannot account for these movements. This is an issue shared by other research on media institutions that aggregate county-level data to the DMA level (Miho, 2018, p. 9), and leads to a slight underestimation of the treatment effect, as a small number of untreated units will be classified as treated and vice-versa.

## 4 Methodology

### 4.1 Tools Used

I use Knitr (Xie, 2014) to integrate statistical calculations into the paper, eliminating the possibility of transcription errors. To ensure the findings are reproducible, I tested the analysis routines using the *testthat* package in R (Wickham, 2011) and the *unittest* module in Python (Van Rossum and Drake, 2009). But I won’t make you take my word that my methods are properly implemented – I provide a Docker image and reproducibility materials to ensure others can replicate the calculations on their own systems (Merkel, 2014; Boettiger, 2015). The result is “one-click reproducibility” (Nüst et al., 2020); readers can reproduce this exact paper with the push of a button from the linked materials. <sup>2</sup>

---

<sup>2</sup>Replication materials available [here](#). By default, the web-scraping does not run, as the data take several days to collect.



## 4.2 Preregistration

To avoid the possibility of fitting hypotheses to the data after results are known, I preregistered my analysis using the Google trends data. Readers can find the preregistration plan in [Appendix C](#).

I have made one deviation from the preregistration. In my preregistration, I describe a different strategy to scale the Google trends data than the one I actually employ. The original strategy was based on a misunderstanding of the format of Google trends data, and does not actually produce the desired measure.

In the analysis I perform, I correct this mistake. I describe the correct scaling procedure in section [3.3.2](#).

## 4.3 Difference-In-Differences Analysis

$$\text{Racially Charged Search Rate}_{tr} = \beta \times \text{Sinclair Present}_{rt} + \lambda_r + \delta_t + \varepsilon_{tr}$$

$$\text{IAT Score}_{tri} = \beta \times \text{Sinclair Present}_{rt} + \lambda_r + \delta_t + \varepsilon_{tri}$$

## 5 Results

Table 1: Fixed Effects Models For Sinclair Aquisition on Google Searches

	<i>Dependent variable:</i>			
	Frequency of Searches for [Word 1]		Frequency of Searches for Words 1-5	
	(1)	(2)	(3)	(4)
Sinclair Present	−0.009 (0.010)	−0.030* (0.016)	−0.001 (0.011)	−0.010 (0.018)
Constant	0.201*** (0.036)	23.678 (18.785)	0.136*** (0.040)	13.430 (21.272)
Year Fixed Effects	Yes	Yes	Yes	Yes
Region Fixed Effects	Yes	Yes	Yes	Yes
Region Time Trends	No	Yes	No	Yes
Observations	3,570	3,570	3,570	3,570
R <sup>2</sup>	0.234	0.349	0.226	0.329
Adjusted R <sup>2</sup>	0.183	0.258	0.173	0.235

*Note:*

\*p<0.1; \*\*p<0.05; \*\*\*p<0.01

	Model 1	Model 2	Model 3	Model 4
(Intercept)	0.28***	0.29***	0.32***	0.32***
	(0.00)	(0.01)	(0.00)	(0.01)
sinclair_presentTRUE	0.00	-0.00	0.00	0.00
	(0.00)	(0.00)	(0.00)	(0.00)
Year Fixed Effects	Yes	Yes	Yes	Yes
Region Fixed Effects	Yes	Yes	Yes	Yes
Region Time Trends	No	Yes	No	Yes
White Respondents Only	No	No	Yes	Yes
Num. obs.	3936939	3936939	2014672	2014672
AIC	7746045.49	7746256.37	3467774.54	3468051.48

\*\*\* $p < 0.001$ ; \*\* $p < 0.01$ ; \* $p < 0.05$

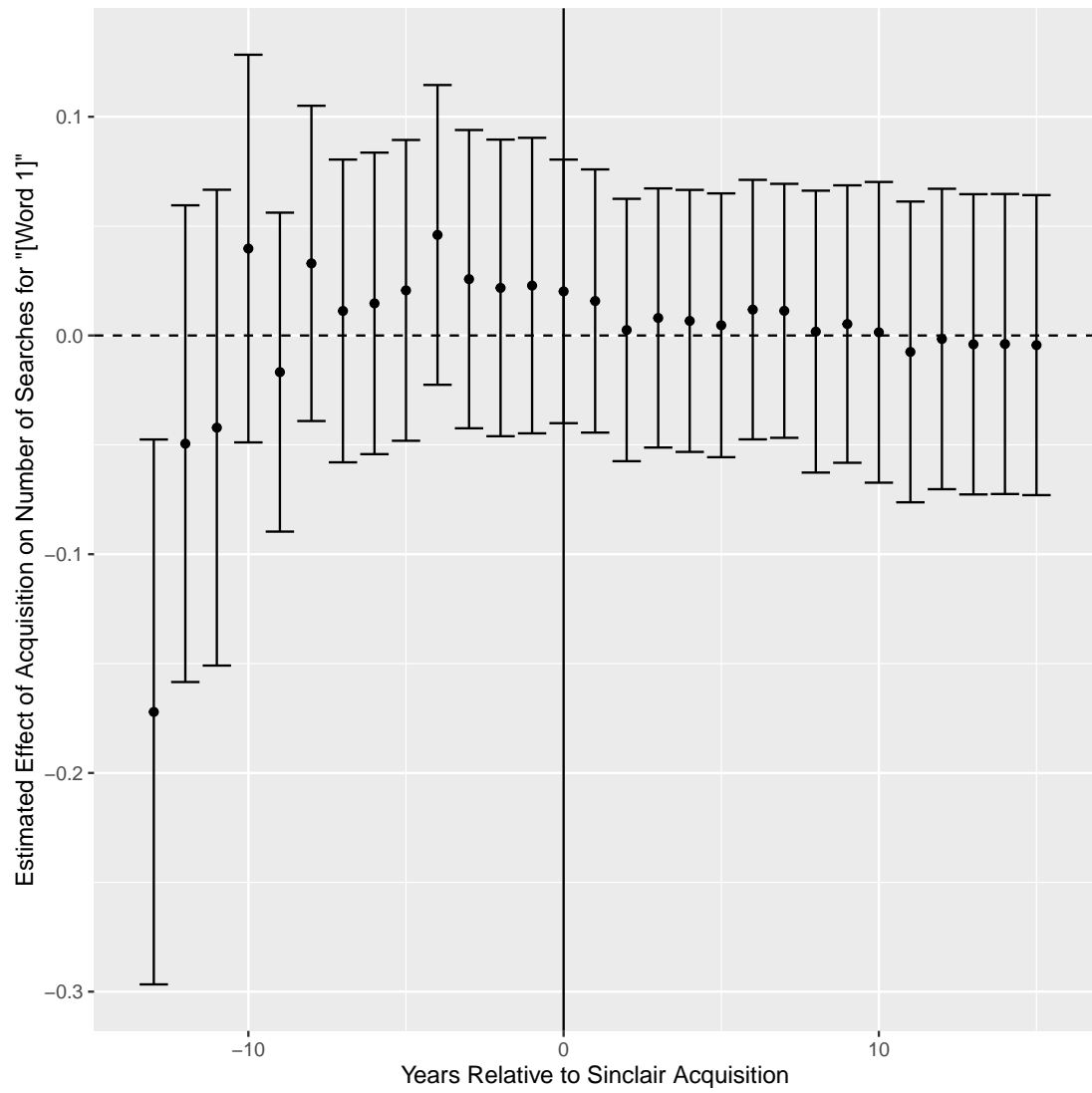
Table 2: Statistical models

## 5.1 Identification Assumption

In this section, I test the indication assumption, the assumption that the treated and control units would have the outcomes if the treatment were absent.

$$\text{Racially Charged Search Rate} = \beta_1(\text{Sinclair Present}) + \beta_2(\text{DMA fixed effects}) + \beta_3(\text{year fixed effects}) + \beta_4(\text{Year} / \text{D})$$

Figure 6: Fixed-Effects Estimates of the effect of Sinclair Acquisition on Searches for “[Word 1] ”



## A Codings for Offensive Words

Table 3: Coding For Offensive Words

Code	Word
Word 1	Nigger
Word 2	Coon
Word 3	Kike
Word 4	Spic
Word 5	Spook

## B References

- Ansolabehere, Stephen (Oct. 2006). “The Orientation of Newspaper Endorsements in U.S. Elections, 1940–2002”. In: *Quarterly Journal of Political Science* 1.4, pp. 393–404.
- Boettiger, Carl (Jan. 2015). “An introduction to Docker for reproducible research”. In: *ACM SIGOPS Operating Systems Review* 49.1, pp. 71–79.
- Burke, Timothy (2018). *How I Made A Dumb Video Making Fun Of Sinclair Broadcasting And Somehow Started A Media War*. URL: <https://deadspin.com/how-i-made-a-dumb-video-making-fun-of-sinclair-broadcas-1825106452> (visited on 03/23/2020).
- Chae, David H., Sean Clouston, Mark L. Hatzenbuehler, et al. (Apr. 2015). “Association between an Internet-Based Measure of Area Racism and Black Mortality”. In: *PLOS ONE* 10.4. Ed. by Hajo Zeeb, e0122963.
- Chae, David H., Sean Clouston, Connor D. Martz, et al. (Feb. 2018). “Area racism and birth outcomes among Blacks in the United States”. In: *Social Science & Medicine* 199, pp. 49–55.
- Cohen, Stephen (2018). *KOMO attacks ‘biased and false news’ in Sinclair-written promos*. URL: <https://www.seattlepi.com/seattlenews/article/KOMO-fake-news-Sinclair-promos-12792032.php> (visited on 03/23/2020).
- Connor, Paul et al. (Jan. 2019). “Income Inequality and White-on-Black Racial Bias in the United States: Evidence From Project Implicit and Google Trends”. In: *Psychological Science* 30.2, pp. 205–222.

- DellaVigna, S. and E. Kaplan (Aug. 2007). “The Fox News Effect: Media Bias and Voting”. In: *The Quarterly Journal of Economics* 122.3, pp. 1187–1234.
- Gentzkow, Matthew (Aug. 2006). “Television and Voter Turnout”. In: *Quarterly Journal of Economics* 121.3, pp. 931–972.
- Gentzkow, Matthew, Jesse M Shapiro, and Michael Sinkinson (Dec. 2011). “The Effect of Newspaper Entry and Exit on Electoral Politics”. In: *American Economic Review* 101.7, pp. 2980–3018.
- GOP Trust PAC (2010). *Breaking Point: Part II*. URL: [https://www.youtube.com/watch?v=vSD9v8K1\\_5s](https://www.youtube.com/watch?v=vSD9v8K1_5s) (visited on 04/20/2020).
- Hill, Andrew (2015). *DMA Boundaries*. URL: [https://team.carto.com/u/andrew/tables/dma\\_boundary/public](https://team.carto.com/u/andrew/tables/dma_boundary/public).
- Isoya, Hidehito and Hiroyuki Yamada (Mar. 2021). “Hidden internet-based measure of discrimination and mental health of Black people”. In: *Social Science & Medicine* 273, p. 113776.
- Kim, Do-Yeong (Mar. 2003). “Voluntary Controllability of the Implicit Association Test (IAT)”. In: *Social Psychology Quarterly* 66.1, p. 83.
- Krumpal, Ivar (Nov. 2011). “Determinants of social desirability bias in sensitive surveys: a literature review”. In: *Quality & Quantity* 47.4, pp. 2025–2047.
- Kuklinski, James H., Michael D. Cobb, and Martin Gilens (1997). “Racial Attitudes and the ”New South””. In: *The Journal of Politics* 59.2, pp. 323–349.
- Ladd, Jonathan McDonald and Gabriel S. Lenz (Apr. 2009). “Exploiting a Rare Communication Shift to Document the Persuasive Power of the News Media”. In: *American Journal of Political Science* 53.2, pp. 394–410.
- Martin, Gregory J. and Joshua McCrain (Feb. 2019). “Local News and National Politics”. In: *American Political Science Review* 113.2, pp. 372–384.
- Merkel, Dirk (2014). “Docker: lightweight linux containers for consistent development and deployment”. In: *Linux journal* 2014.239, p. 2.
- Miho, Antonela (2018). “Small screen, big echo? Estimating the political persuasion of local television news bias using the Sinclair Broadcasting Group as a natural experiment”. In: *SSRN Electronic Journal*.
- Morris, Jonathan S. (July 2005). “The Fox News Factor”. In: *Harvard International Journal of Press/Politics* 10.3, pp. 56–79.

- Mullainathan, Sendhil and Andrei Shleifer (Aug. 2005). “The Market for News”. In: *American Economic Review* 95.4, pp. 1031–1053.
- Nüst, Daniel et al. (Nov. 2020). “Ten simple rules for writing Dockerfiles for reproducible data science”. In: *PLOS Computational Biology* 16.11. Ed. by Scott Markel, e1008316.
- Pleat, Zachary (2020a). <https://www.mediamatters.org/sinclair-broadcast-group/sinclair-program-suggests-vigilantes-are-needed-segment-promoted-warnings>. URL: <https://www.mediamatters.org/sinclair-broadcast-group/sinclair-program-suggests-vigilantes-are-needed-segment-promoted-warnings>.
- (2020b). *These seven national Sinclair Broadcast news segments on the Floyd protests didn’t mention police violence against protesters*. URL: <https://www.mediamatters.org/black-lives-matter/these-seven-national-sinclair-broadcast-news-segments-floyd-protests-didnt>.
- Pleat, Zachary and Rob Savillo (2020). *Sinclair reporters pushed debunked myths in national segments on police violence against Black Americans*. URL: <https://www.mediamatters.org/black-lives-matter/sinclair-reporters-pushed-debunked-myths-national-segments-police-violence>.
- Scherer, Dana A. (2018). *Federal Communications Commission (FCC) Media Ownership Rules*. URL: <https://fas.org/sgp/crs/misc/R45338.pdf>.
- Sood, Gaurav (2016). *Geographic Information on Designated Media Markets*. Version V9.
- Stephens-Davidowitz, Seth (Oct. 2014). “The cost of racial animus on a black candidate: Evidence using Google search data”. In: *Journal of Public Economics* 118, pp. 26–40.
- Van Rossum, Guido and Fred L. Drake (2009). *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace.
- Wetts, Rachel and Robb Willer (May 2018). “Privilege on the Precipice: Perceived Racial Status Threats Lead White Americans to Oppose Welfare Programs”. In: *Social Forces* 97.2, pp. 793–822.
- Wickham, Hadley (2011). “testthat: Get Started with Testing”. In: *The R Journal* 3, pp. 5–10.
- Willer, Robb, Matthew Feinberg, and Rachel Wetts (2016). “Threats to Racial Status Promote Tea Party Support Among White Americans”. In: *SSRN Electronic Journal*.

- Xie, Yihui (2014). “knitr: A Comprehensive Tool for Reproducible Research in R”. In: *Implementing Reproducible Computational Research*. Ed. by Victoria Stodden, Friedrich Leisch, and Roger D. Peng. ISBN 978-1466561595. Chapman and Hall/CRC.
- Xu, Kaiyuan et al. (2020). *Project Implicit: About the IAT*. URL: <https://implicit.harvard.edu/implicit/iatdetails.html> (visited on 02/18/2021).
- (Jan. 2021). *Project Implicit Demo Website Datasets*.

## C Preregistration Document



## C.1 Pre-registration

### Pre-Registration for Undergrad Dissertation

Beniamino Green

March 15, 2021

*This paper uses Google search terms containing offensive language as a proxy measure for racial animus. Following the practice of similar works (Stephens-Davidowitz 2014; Chae, Clouston, Hatzenbuehler, et al. 2015; Chae, Clouston, Martz, et al. 2018; Isoya and Yamada 2021), I use coded language to refer to these terms. The words themselves can be found in Table ??*

## 1 Introduction

Can media coverage influence racial resentment? Empirical studies suggest that when white Americans understand welfare policies to threaten their privileged status in the U.S. Social hierarchy, their resentment of minorities increases and their support for welfare decreases (Willer, Feinberg, and Wetts 2016; Wetts and Willer 2018). This tendency seems to be weaponized by conservative media institutions and politicians, who seem to encourage racial animus to erode support for social programs, as in the infamous example of the “Welfare Queen” narrative, a racial stereotype employed to undercut support for the Aid to Families with Dependent Children (AFDC). However, the link between traditional media coverage and racial resentment has not been extensively studied.

I propose a study exploiting the expansion of the Sinclair Media Network from 2004-2021 to understand how conservative media messaging impacts racial resentment in a media market. The expansion of Sinclair Media during this period provides the basis for a difference-in-differences analysis estimating the effect of Sinclair Media purchasing a station on racial animus in an area. I propose using data from Google search trends in an area as a proxy for racial resentment, a strategy that has already been used to measure racial resentment in the context of public health (Chae, Clouston, Martz, et al. 2018; Chae, Clouston, Hatzenbuehler, et al. 2015) and elections research (Stephens-Davidowitz 2014).

## 2 Background

### 2.1 Sinclair Broadcast Group

In this paper, I suggest exploiting the expansion of Sinclair Network over the period of 2004-2021. During this, the Sinclair Broadcast Group sold or purchased stations in 67 media markets. I use the same strategy to track the expansion of Sinclair Media outlets employed by Miho 2018, namely, by extracting a record of the stations Sinclair Owns at the end of each financial year from their SEC form 10-K filings. These filings have an advantage over the typical sets data used to track network expansion maintained by the Nielsen Corporation as they are publicly available, so the findings can be easily reproduced.

Previous research has demonstrated that Sinclair acquisition of a network is associated with a sharp rightwards shift in its coverage (Martin and McCrain 2019). However, a quantitative analysis of the effect of Sinclair ownership on coverage of racial issues has not been conducted. There is significant evidence to suggest that Sinclair ownership does push stations to run more

## C.2 Pregistration

racially conservative stories than they otherwise would. This year, the Sinclair corporation drew ire for a series of “must-run” segments on police violence following the murder of George Floyd pushing the “black-on-black violence” canard and advocating for a military response to the protests (Pleat and Savillo 2020; Pleat 2020).

In analyzing the effects of news coverage on racial animus, it might be natural to examine the expansion of Fox News, which occupies a position in the public consciousness as among the most conservative stations on racial issues, and is the most trusted media outlet among Republican and Republican-leaning respondents in many polls (Mark Jurkowitz and Walker 2020). However, I choose to use Sinclair over Fox News Stations for two reasons. First, Fox News’ expansion strategy has involved purchasing a larger stations. As television companies in the US can only expand until they broadcast to 39% of U.S. households (Scherer 2018), Fox News has been able to buy fewer stations than Sinclair, which entails a smaller sample size of stations that changed ownership. Second, Fox News’ expansion primarily happened before 2004, the first year for which Google Trends data is available, which further limits the sample size of stations which changed ownership for which there is data on racial animus.

### 2.2 Google Trends Data

Following Stephens-Davidowitz 2014, I use Google trends data as a proxy measure for racial animus. Specifically, I use trend data for the searches for the terms “[Word 1] ” and “[Word 1]’s ”.

This measurement approach has an advantage over traditional survey-based measures of racial animus in that it is less subject to a social desirability bias; “Google searchers are online and likely alone, both of which make it easier to express socially taboo thoughts (Kreuter et al., 2009)” (ibid., p. 26). Further, it provides a high-resolution set of data that would be prohibitively expensive to collect from a traditional survey, especially given that the difference-in-differences approach requires a repeated survey comparable across multiple time periods.

A pressing concern is that Google searches for “[Word 1](s) ” may not actually capture the extent of racial bias in an area, but simply reflect users learning about the term. In fact, “definition of [Word 1] ” and “what does [Word 1] mean” are both among the top 5 search queries related to the “[Word 1] ”. These queries suggest that many who search for the term are searching out of curiosity to investigate the term.

This is not to suggest that searches for the term do not capture any variation in racial animus: among the top 10 most searched related queries to the term [Word 1]’s are “I don’t like [Word 1]’s ,” “fuck the [Word 1]’s ,” and “ship those [Word 1]’s back.”

The concern that Google Searches for racial slurs may largely reflect curiosity about the term is valid. Subject to data availability, I suggest controlling for the frequency of searches for the definitions of these terms in each area, to try and isolate the effect of Sinclair media entering a market on searches expressing “hardcore” racial animus rather than curiosity.

## 3 Methodology

### 3.1 Scaling Google Trends Data

#### 3.1.1 Between Regions

Google trends data provide a sample of the searches made by Google users over a certain time frame in a region. The popularity of a search in a region at a certain time is expressed as a search score, calculated as the following:

$$\text{Search Score}_{ijk} = \left[ \frac{\text{number of searches for } i \text{ during time } j \text{ in region } k}{\text{maximum number of searches for } i \text{ over any month in region } k} \right] \times 100$$

## C.3 Pregistration

Where

- $i$  is the search term in question
- $j$  is the time period analyzed (a given year)
- $k$  is the region surveyed

This poses a problem for a difference-in-difference approach, as the maximum number of searches for a word in any time period changes between areas. As the maximum monthly searches for a term differs between regions, A score of 50 in one region might correspond to 500 searches per capita, while the same score might correspond to 5000 searches per capita in another area.

Accordingly, I scale searches between regions to make them directly comparable by multiplying the search score in a given area for a term by the average frequency with which that term is searched in that region when compared to other regions, a separate set of data also available from Google Trends. This has the effect of standardizing the search scores, so that any given score always corresponds to the same number of searches per capita.

$$\text{Scaled Search Score}_{ijk} = \text{Search Score}_{ijk} \times \text{average frequency of searches for } i \text{ in region } k$$

### 3.1.2 Between Search Terms

Past studies have only used search data for one or two searches (Stephens-Davidowitz 2014, p. 26), primarily “[Word 1](s)”. This approach makes for easily interpretable results, but leaves open the possibility that the findings simply reflect idiosyncratic changes in searches these terms rather than changes in racial animus as a whole. I suggest, subject to data availability, using a larger pool of search terms (registered before running tests) to try and understand whether findings are consistent across a larger set of searches that express racial animus.

Currently, Google Trends only allows for the direct comparison of up to five searches at once. When these queries are performed, the search score for each search is expressed as a percent of the frequency of searches in the most popular region. I solve this problem by using several linked sets of comparisons. By running several sets of 5 queries with a shared term (in this case, “smart”), I can scale the searches so they are directly comparable. In the example below, this entails multiplying the score of words 5-9 by 2, so that the scores are on the same scale - a percent of the frequency of searches for “fruit” in DMA 803 (the term / region combination searched most frequently). This makes the scores directly comparable.

Table 1: Example Raw Data From Google Trends

DMA Code	shoe	fruit	horse	cup	smart
803	23	100	5	10	27
616	26	86	3	15	45
617	32	94	6	24	31

Search trends data for words 1-5

DMA Code	smart	pen	waltz	gnome	boots
803	14	32	5	3	100
616	22	54	13	2	88
617	16	50	12	1	96

Search trends data for words 5-9

### 3.2 Difference in Differences Analysis

To estimate the difference-in-difference models, I will use the following model:

$$\text{Racially Charged Search Rate} = \beta_1(\text{Sinclair Present}) + \beta_2(\text{DMA fixed effects}) + \beta_3(\text{year fixed effects}) +$$

## C.4 Pregristration

I will also estimate the effect using a Poisson / negative binomial regression if the counts of searches for the terms used are over-dispersed, as performed in Chae, Clouston, Hatzenbuehler, et al. 2015.

$$\log(\text{Racially Charged Search Rate}) = \beta_1(\text{Sinclair Present}) + \beta_2(\text{DMA fixed effects}) + \beta_3(\text{year fixed effects}) +$$

## References

- Chae, David H., Sean Clouston, Mark L. Hatzenbuehler, et al. (Apr. 2015). “Association between an Internet-Based Measure of Area Racism and Black Mortality”. In: *PLOS ONE* 10.4. Ed. by Hajo Zeeb, e0122963.
- Chae, David H., Sean Clouston, Connor D. Martz, et al. (Feb. 2018). “Area racism and birth outcomes among Blacks in the United States”. In: *Social Science & Medicine* 199, pp. 49–55.
- Isoya, Hidehito and Hiroyuki Yamada (Mar. 2021). “Hidden internet-based measure of discrimination and mental health of Black people”. In: *Social Science & Medicine* 273, p. 113776.
- Mark Jurkowitz Amy Mitchell, Elisa Shearer and Mason Walker (Jan. 2020). *U.S. Media Polarization and the 2020 Election: A Nation Divided*. URL: <https://www.journalism.org/2020/01/24/u-s-media-polarization-and-the-2020-election-a-nation-divided/>.
- Martin, Gregory J. and Joshua McCrain (Feb. 2019). “Local News and National Politics”. In: *American Political Science Review* 113.2, pp. 372–384.
- Miho, Antonela (2018). “Small screen, big echo? Estimating the political persuasion of local television news bias using the Sinclair Broadcasting Group as a natural experiment”. In: *SSRN Electronic Journal*.
- Pleat, Zachary (2020). *These seven national Sinclair Broadcast news segments on the Floyd protests didn't mention police violence against protesters*. URL: <https://www.mediamatters.org/black-lives-matter/these-seven-national-sinclair-broadcast-news-segments-floyd-protests-didnt>.
- Pleat, Zachary and Rob Savillo (2020). *Sinclair reporters pushed debunked myths in national segments on police violence against Black Americans*. URL: <https://www.mediamatters.org/black-lives-matter/sinclair-reporters-pushed-debunked-myths-national-segments-police-violence>.
- Scherer, Dana A. (Oct. 2018). “Federal Communications Commission (FCC) Media Ownership Rules”. In: *Congressional Research Service Reports*.
- Stephens-Davidowitz, Seth (Oct. 2014). “The cost of racial animus on a black candidate: Evidence using Google search data”. In: *Journal of Public Economics* 118, pp. 26–40.
- Wetts, Rachel and Robb Willer (May 2018). “Privilege on the Precipice: Perceived Racial Status Threats Lead White Americans to Oppose Welfare Programs”. In: *Social Forces* 97.2, pp. 793–822.
- Willer, Robb, Matthew Feinberg, and Rachel Wetts (2016). “Threats to Racial Status Promote Tea Party Support Among White Americans”. In: *SSRN Electronic Journal*.

## A Codings for Offensive Words

Code	Word
Word 1	nigger

## D Code

To fulfill requirements that the code for this work is scanned by turnitin, I append the code used to make this report below. For human readers interested in the code, I strongly recommend the [replication materials](#).

### D.1 Master Replication Script

```
1 #!/bin/sh
2
3 echo "Running Unit Tests" &&
4   python -m unittest &&
5   echo "testthat::test_dir("tests/testthat")" | R -q --vanilla ||
6   echo "Unit Tests Failed"
7
8 Rscript ./01_clean_sinclair_data.R && # python 02_webscrape.py &&
9   Rscript ./03_clean_search_data.R &&
10  Rscript ./04_clean_iat_data.R &&
11  Rscript ./05_models.R
```

### D.2 Cleaning Sinclair Data

```
1 #!/usr/bin/Rscript
2 # Load Libraries for Analysis
3 library(tidyverse)
4 library(lubridate)
5 # Load utility functions
6 source("utils.R")
7
8 # Create tidy dataframe of years / markets in which sinclair was present
9 sinclair_raw <- tibble(filename = list.files(path = "data/sinclair_data/", full.names = T,
10   pattern = "*.csv")) %>%
11   mutate(
12     year = str_extract(filename, "[0-9]+") %>% as.double(),
13     data = filename %>% map(~ read_csv(.x, locale = locale(encoding = "UTF-8"))),
14     data = map(data, ~ rename_with(.x, ~ gsub("\n", "", .x))),
15     data = map(data, ~ .x %>% rename_with(tolower)),
16     data = map(data, ~ .x %>% dplyr::select(market))
17   ) %>%
18   unnest(data) %>%
19   select(-filename) %>%
20   mutate(
21     sinclair_present = T,
22     market = tolower(gsub("[^A-z]", "", market))
23   ) %>%
24   filter(market != "")
25
26 # Loads in Translation dictionary of market names in the Sinclair
27 # dataset to DMA codes
28 sinclair_codes <- read_csv("data/sinclair_names.csv")
29
30 # checks that the sinclair codes table has all markets from the sinclair
31 # present dataset
32 stopifnot(
33   all(sinclair_raw$market %in% sinclair_codes$market),
34   all(sinclair_codes$market %in% sinclair_raw$market)
35 )
36
37 # Adds codes to sinclair present dataset
38 sinclair_present <- sinclair_raw %>%
39   inner_join(sinclair_codes) %>%
40   select(-market)
41
42 # checks no observations were lost in merge
43 stopifnot(nrow(sinclair_raw) == nrow(sinclair_present))
44 # checks there are no duplicate rows
45 stopifnot(!any(duplicated(sinclair_present)))
46
47 # Loads in Translation dictionary of DMA codes to Standardized names
```

```

47 dma_names <- read_csv("data/dma_list.csv")
48 # Checks there are 210 DMA marketes in lookup table
49 stopifnot(nrow(dma_names) == 210)
50
51 # Fills in data to include stations where sinclair was not present
52 sinclair_data <- sinclair_present %>%
53   complete(code = dma_names$code, year = 2004:2020) %>%
54   arrange(code) %>%
55   replace_na(list(sinclair_present = F))
56
57
58 # checks there is an observation for every year and in the dataset
59 stopifnot({
60   sinclair_data %>%
61     anti_join(expand.grid(code = dma_names$code, year = 2004:2020)) %>%
62     nrow() == 0
63 })
64 # checks there are no duplicate rows
65 stopifnot(!any(duplicated(sinclair_data)))
66
67 sinclair_data <- sinclair_data %>%
68   group_by(code) %>%
69   mutate(years_before = years_before(sinclair_present))
70
71 # write out to csv file
72 write_csv(sinclair_data, "../data/clean_sinclair_data.csv")

```

## D.3 Web Scraping

```

1  #!/usr/bin/env python
2  from scaling import recover_scores, repair_matrix
3  from in_region import in_region, to_wide
4  from between_regions import create_v_df, between_region
5  import tqdm as tqdm
6  import pickle as pkl
7  import pandas as pd
8
9  with open("data/dma_abbreviations.pkl", "rb") as f:
10     dmas = pkl.load(f)
11
12 terms = ["nigger", "coon", "kike", "spic", "spook"]
13
14 for i, term in tqdm.tqdm(enumerate(terms)):
15     in_region_dfs = tuple(
16         in_region(term, dma, True, timeframe="all") for dma in tqdm.tqdm(dmas))
17     wide_dfs = map(to_wide, in_region_dfs)
18     h_df = pd.concat(wide_dfs).sort_index()
19
20     dfs = tuple(
21         create_v_df([term], year) for year in tqdm.tqdm(range(2004, 2021)))
22     v_df = pd.concat(dfs, axis=1).sort_index()
23
24     h_df = h_df.drop("2021", axis=1)
25     v_matrix = v_df.to_numpy(copy=True)
26     h_matrix = h_df.to_numpy(copy=True)
27
28     gen1 = recover_scores(h_matrix, v_matrix)
29     final = repair_matrix(h_matrix, v_matrix, gen1)
30     final = pd.DataFrame(final)
31     final.columns = h_df.columns
32     final.index = h_df.index
33
34     final.to_csv(f"data/scaled_word_{i+1}.csv")
35
36 old_df = None
37 for year in range(2004, 2021):
38
39     timeframe = f"{year}-01-01 {year}-12-31"
40     df = between_region(terms,
41                         censor=False,
42                         timeframe=timeframe,
43                         geo="US",
44                         gprop="")
45     df['year'] = year
46
47     old_df = pd.concat([old_df, df], axis=0)

```

```

48 old_df.rename(columns={
49   old_df.columns[0]: "word1_weight",
50   old_df.columns[1]: "word2_weight",
51   old_df.columns[2]: "word3_weight",
52   old_df.columns[3]: "word4_weight",
53   old_df.columns[4]: "word5_weight"
54 },
55 ,
56   inplace=True)
57
58 old_df.to_csv("data/between_region_comparisons.csv")

```

## D.4 Cleaning Google Search Data

```

1 library(tidyverse)
2
3 # Get dataframe of searches by DMA
4 scores <- tibble(file = list.files(path = "data", pattern = "scaled_word.*", full.names = T))
5   %>%
6   mutate(data = map(file, read_csv)) %>%
7   mutate(data = map2(data, file, ~ .x %>% mutate(term = str_extract(.y, "word_[0-9]")))) %>%
8   unnest(data) %>%
9   pivot_longer(as.character(2004:2020), names_to = "year", values_to = "score") %>%
10  select(~file) %>%
11  pivot_wider(names_from = term, values_from = score) %>%
12  mutate(year = as.numeric(year))
13 stopifnot(nrow(scores) == 3570)
14
15 # Put different search terms on the same scale
16 scales <- read_csv("data/between_region_comparisons.csv")
17 scaled_data <- full_join(scores, scales) %>%
18   mutate(
19     sword_1 = word_1 * word1_weight/100,
20     sword_2 = word_2 * word2_weight/100,
21     sword_3 = word_3 * word3_weight/100,
22     sword_4 = word_4 * word4_weight/100,
23     sword_5 = word_5 * word5_weight/100,
24     score = sword_1 + sword_2 + sword_3 + sword_4 + sword_5
25   )
26 stopifnot(nrow(scaled_data) == 3570)
27
28 sinclair_data <- read_csv("../data/clean_sinclair_data.csv")
29 full_data <- full_join(sinclair_data, scaled_data)
30
31 stopifnot(nrow(full_data) == 3570)
32
33 full_data %>% write_csv("../data/clean_search_data.csv")

```

## D.5 Cleaning IAT Data

```

1 #!/usr/bin/Rscript
2 library(tidyverse)
3
4 #Load in all years of IAT data, consolidate race into a single column
5 full_iat_data <- list.files(path = "data", pattern = "race.*.csv", full.names = T) %>%
6   map_dfr(read_csv) %>%
7   tibble() %>%
8   unite("race", raceomb:raceomb_002, sep = "", na.rm = T, remove = T) %>%
9   rename_with(tolower) %>%
10  group_by(state)
11
12 #Get estimates of bias by state
13 full_iat_data %>%
14   filter(race == 6, ethnicityomb == 2) %>%
15   summarize(iat = mean(d_biep.white_good_all, na.rm = T), sbp5 = mean(sbp5, na.rm = T)) %>%
16   select(state, iat, sbp5) %>%
17   write_csv("../data/iat_state_data.csv")
18
19 #Read in mapping of DMA codes to state codes
20 dma_to_zip <- read_csv("data/DMA-zip.csv") %>%
21   select(~ZIPCODE) %>%
22   distinct()
23
24 #Read in mapping of DMA codes to state codes

```

```

25 state_to_no <- read_csv("data/state_info.csv")
26
27 #Create FIPS county code column
28 iat_by_county <- full_join(state_to_no, full_iat_data) %>%
29   mutate(
30     countyno = countyno %>% str_pad(3, pad = "0"),
31     FIPS = str_pad(paste0(state.no, countyno), 5, pad = "0")
32   )
33
34 #Test column / join was created correctly without dropping any observations
35 stopifnot(nrow(iat_by_county) == nrow(full_iat_data))
36 rm(state_to_no, full_iat_data)
37
38 # Add DMA column to IAT data
39 iat_by_dma <- iat_by_county %>% right_join(dma_to_zip)
40 rm(iat_by_county, dma_to_zip)
41
42 # make column names pretty and conforming with other datasets
43 iat_by_dma <- rename_with(iat_by_dma, ~ tolower(gsub(" ", "_", .x, fixed = TRUE))) %>%
44   rename(code = dma_code) %>%
45   mutate(code = as.numeric(code))
46
47 sinclair_data <- read_csv("../data/clean_sinclair_data.csv")
48
49 clean_iat_data <- right_join(sinclair_data, iat_by_dma)
50 stopifnot(nrow(iat_by_dma) == nrow(clean_iat_data))
51
52 save(clean_iat_data, file = "../data/clean_iat_data.Rdata")

```

## D.6 Running Models

```

1  #!/usr/bin/Rscript
2  library(tidyverse)
3  library(biglm)
4
5  search_data <- read_csv("../data/clean_search_data.csv")
6  stopifnot(nrow(search_data) == 3570)
7
8  print("Estimating Model 1 / 12")
9  model_1 <- lm(word_1 ~ as.factor(year) + as.factor(code) + sinclair_present, data = search_
10    data)
11
12 print("Estimating Model 2 / 12")
13 model_2 <- lm(word_1 ~ as.factor(year) + as.factor(code) + as.factor(code):year + sinclair_
14   present, data = search_data)
15
16 print("Estimating Model 3 / 12")
17 model_3 <- lm(word_1 ~ as.factor(year) + as.factor(code) + as.factor(years_before), data =
18   search_data)
19
20 print("Estimating Model 4 / 12")
21 model_4 <- lm(score ~ as.factor(year) + as.factor(code) + sinclair_present, data = search_
22   data)
23
24 print("Estimating Model 5 / 12")
25 model_5 <- lm(score ~ as.factor(year) + as.factor(code) + as.factor(code):year + sinclair_
26   present, data = search_data)
27
28 print("Estimating Model 6 / 12")
29 model_6 <- lm(score ~ as.factor(year) + as.factor(code) + as.factor(years_before), data =
30   search_data)
31
32 # loads clean_iat_data
33 load("../data/clean_iat_data.Rdata")
34
35 clean_iat_data <- clean_iat_data %>%
36   mutate(
37     code = as.factor(code),
38     factor_year = as.factor(year),
39     years_before = as.factor(years_before)
40   ) %>%
41   as.data.frame()
42
43 print("Estimating Model 7 / 12")
44 chunks_7 <- split(clean_iat_data, (as.numeric(rownames(clean_iat_data)) - 1) %/% 500000)
45 chunks_7[[1]] <- biglm(d_biep.white_good_all ~ factor_year + code + sinclair_present, data =

```



```

    chunks_7[[1]])
40 model_7 <- reduce(chunks_7, update)
41 rm(chunks_7)
42
43 print("Estimating Model 8 / 12")
44 chunks_8 <- split(clean_iat_data, (as.numeric(rownames(clean_iat_data)) - 1) %% 500000)
45 chunks_8[[1]] <- biglm(d_biep.white_good_all ~ factor_year + code + code:year + sinclair_
    present, data = chunks_8[[1]])
46 model_8 <- reduce(chunks_8, update)
47 rm(chunks_8)
48
49 print("Estimating Model 9 / 12")
50 chunks_9 <- split(clean_iat_data, (as.numeric(rownames(clean_iat_data)) - 1) %% 500000)
51 chunks_9[[1]] <- biglm(d_biep.white_good_all ~ factor_year + code + years_before, data =
    chunks_9[[1]])
52 model_9 <- reduce(chunks_9, update)
53 rm(chunks_9)
54
55 iat_white_nonhisp <- clean_iat_data %>%
56   filter(race == 6, ethnicityomb == 2) %>%
57   as.data.frame()
58
59 print("Estimating Model 10 / 12")
60 chunks_10 <- split(iat_white_nonhisp, (as.numeric(rownames(iat_white_nonhisp)) - 1) %%
    500000)
61 chunks_10[[1]] <- biglm(d_biep.white_good_all ~ factor_year + code + sinclair_present, data =
    chunks_10[[1]])
62 model_10 <- reduce(chunks_10, update)
63 rm(chunks_10)
64
65 print("Estimating Model 11 / 12")
66 chunks_11 <- split(iat_white_nonhisp, (as.numeric(rownames(iat_white_nonhisp)) - 1) %%
    500000)
67 chunks_11[[1]] <- biglm(d_biep.white_good_all ~ factor_year + code + code:year + sinclair_
    present, data = chunks_11[[1]])
68 model_11 <- reduce(chunks_11, update)
69 rm(chunks_11)
70
71 print("Estimating Model 12 / 12")
72 chunks_12 <- split(iat_white_nonhisp, (as.numeric(rownames(iat_white_nonhisp)) - 1) %%
    500000)
73 chunks_12[[1]] <- biglm(d_biep.white_good_all ~ factor_year + code + years_before, data =
    chunks_12[[1]])
74 model_12 <- reduce(chunks_12, update)
75 rm(chunks_12)
76
77 save(model_1, model_2, model_3, model_4, model_5, model_6, model_7, model_8, model_9, model_
    10, model_11, model_12, file = "../data/models.Rdata")

```

## D.7 Code in This Document

```

1 ## ----echo=FALSE, warn=FALSE, include=FALSE-----
2 # Set options to knit this document, load libraries
3 knitr::opts_chunk$set(out.width="100%", message=F, cache=F, echo=F, warn=F, include=F)
4 library(tidyverse)
5 library(sf)
6 library(broom)
7 library(stargazer)
8
9
10 ## -----
11 # Load and create data to show markets Sinclair has expanded to / moved out of
12 sinclair_expansions <- read_csv("../data/clean_sinclair_data.csv") %>%
13   nest(-code) %>%
14   mutate(
15     any_true = data %>% map_lgl(~any(!.$sinclair_present)),
16     any_false = data %>% map_lgl(~any(!.$sinclair_present)),
17     changed = any_true & any_false
18   )
19
20
21 dma_boundaries <- st_read("../data/dma_boundaries/dma_boundary.shp")
22 dma_boundaries <- merge(dma_boundaries, sinclair_expansions, by.x="dma0", by.y="code")
23
24
25 ## ----include = T-----

```

```

26 #Plot markets Sinclair has expanded to / moved out of
27 ggplot() +
28   geom_sf(data=dma_boundaries, aes(fill=changed), alpha=.5) +
29   scale_fill_manual(values = c(NA, "blue")) +
30   theme_void() +
31   theme(legend.pos = "none")
32
33
34 ## ----include = T, warn = F-----
35 #Plot markets Sinclair has expanded to / moved out of
36 sinclair_expansions %>%
37   unnest(cols=c(data)) %>%
38   unite("fill", changed:sinclair_present, remove = F) %>%
39   mutate(code = fct_reorder2(as.factor(code), as.factor(changed), as.factor(sinclair_present))
40   ) %>%
41   ggplot(aes(year, code, fill=sinclair_present)) +
42   geom_raster() +
43   scale_y_discrete(breaks=c()) +
44   scale_fill_discrete(labels = c("Untreated", "Treated")) +
45   theme(legend.title = element_blank()) +
46   xlab("Year") +
47   ylab("Media Market")
48
49 ## ----include=T, warn=FALSE-----
50 library(ggrepel)
51
52 state_iat_data <- read_csv("../data/iat_state_data.csv")
53 word1_state_data <- read_csv("../data/word1_all_time.csv")
54
55 state_iat_searches <- full_join(state_iat_data, word1_state_data) %>%
56   mutate(sbp5 = 5 - sbp5) %>%
57   drop_na()
58
59
60 ## ----include=T-----
61 ggplot(state_iat_searches, aes(word1, iat, label=state)) +
62   geom_text_repel() +
63   geom_smooth(method="lm") +
64   xlab("Frequency of Google Searches for [Word 1]") +
65   ylab("Scaled IAT Score of White Respondents")
66
67
68 ## ----include=T-----
69 ggplot(state_iat_searches, aes(word1, sbp5, label=state)) +
70   geom_text_repel() +
71   geom_smooth(method="lm") +
72   xlab("Frequency of Google Searches for [Word 1]") +
73   ylab("Scaled Responses to Prompt 5")
74
75
76 ## -----
77
78 load("../data/models.Rdata")
79
80
81 ## ----include=TRUE, results="asis"-----
82 stargazer(model_1, model_2, model_4, model_5,
83   omit=c('~as\\.factor\\(year\\)[0-9]{4}$',
84     '~code[0-9]{3}$',
85     '~code[0-9]{3}\\:year$',
86     "code"
87   ),
88   title = "Fixed Effects Models For Sinclair Aquisition on Google Searches",
89   dep.var.labels = c("Frequency of Searches for \\wone", "Frequency of Searches for Words 1-5"),
90   covariate.labels = c("Sinclair Present", "Constant"),
91   add.lines =
92     list(
93       c("Year Fixed Effects", "Yes", "Yes", "Yes", "Yes"),
94       c("Region Fixed Effects", "Yes", "Yes", "Yes", "Yes"),
95       c("Region Time Trends", "No", "Yes", "No", "Yes")),
96   omit.stat=c("LL", "ser", "f"),
97   table.placement="H")
98
99
100 ## -----

```

```

101 library(bigm)
102 library(texreg)
103 load("../data/models.Rdata")
104
105
106 ## ----include=TRUE, results="asis"-----
107 texreg(list(model_7, model_8, model_10, model_11),
108       custom.gof.rows =
109         list(
110           "Year Fixed Effects" = c("Yes", "Yes", "Yes", "Yes"),
111           "Region Fixed Effects" = c("Yes", "Yes", "Yes", "Yes"),
112           "Region Time Trends" = c("No", "Yes", "No", "Yes"),
113           "White Respondents Only" = c("No", "No", "Yes", "Yes")
114         ),
115       float.pos = "H",
116       omit.coef="code|year")
117
118
119 ## ----include =T, warn = FALSE-----
120 model_3 %>%
121   tidy() %>%
122   na.omit() %>%
123   filter(grepl("years_before", term)) %>%
124   mutate(term = as.numeric(gsub("[^0-9\\-]+", "", term))) %>%
125   ggplot(aes(x = term, y = estimate)) +
126   geom_point() +
127   geom_errorbar(aes(ymin = estimate - 1.96 * std.error, ymax = estimate + 1.96 * std.error))
128   +
129   geom_hline(aes(yintercept=0), linetype=2) +
130   geom_vline(aes(xintercept=0)) +
131   xlab("Years Relative to Sinclair Acquisition") +
132   ylab("Estimated Effect of Acquisition on Number of Searches for \"[Word 1]\"")
133
134 ## ----echo=FALSE, include=T-----
135 # Pull R code out of this document to put into the appendix
136 code <- knitr::purl("diss.Rnw", quiet=T)

```

## D.8 Utility Functions - R

```

1 #!/usr/bin/Rscript
2 censor_string <- function(string) {
3   substr(string, 2, nchar(string) - 1) <- paste0(rep("_", nchar(string) - 2), collapse = "")
4   names(string) <- NULL
5   return(string)
6 }
7 censor_string <- Vectorize(censor_string, USE.NAMES=F)
8
9 years_before <- function(bool) {
10   if (any(bool)) {
11     out <- numeric(length = length(bool))
12     start <- min(which(bool))
13
14     before <- (seq(start, 1) - 1) * -1
15     if (start != length(bool)) {
16       after <- seq(1, length(bool) - start)
17     } else {
18       after <- c()
19     }
20
21     out <- c(before, after)
22   } else {
23     out <- rep(-99, length(bool))
24   }
25   return(out)
26 }
27
28 }

```

## D.9 Utility Functions - Python

### D.9.1 Get Between-Region Google Search Data

```

1 #!/usr/bin/env python
2 from pytrends.request import TrendReq

```

```

3 from utils import censor_string
4 import pandas as pd
5
6
7 def between_region(query, censor, **kwargs):
8     pytrends = TrendReq(hl='en-US', tz=360)
9     pytrends.build_payload(query, cat=0, **kwargs)
10    df = pytrends.interest_by_region(resolution='DMA',
11                                    inc_low_vol=True,
12                                    inc_geo_code=True)
13    df = df.set_index("geoCode")
14    df.index.name = 'code'
15    if censor:
16        df = df.rename(censor_string, axis="columns")
17    return (df)
18
19 def between_reigion_many(iterable, censor, **kwargs):
20     iterable = iter(iterable)
21     chunk1 = next(iterable)
22     df1 = between_region(chunk1, censor, **kwargs)
23
24     for chunk in iterable:
25         chunk2 = chunk
26         shared = list(set(chunk1) & set(chunk2))[0]
27         if censor:
28             shared = censor_string(shared)
29         df2 = between_region(chunk2, censor, **kwargs)
30
31         mean1 = df1[shared].mean()
32         mean2 = df2[shared].mean()
33         normaliation_factor = mean1 / mean2
34
35         df2 = df2 * normaliation_factor
36         df2 = df2.drop(columns=[shared])
37
38         df1 = df1.join(df2)
39
40         chunk1 = chunk2
41
42     return (df1)
43
44
45 def create_v_df(term, year):
46     timeframe = f"{year}-01-01 {year}-12-31"
47     df = between_region(term, False, timeframe=timeframe, geo="US")
48     df = df.rename(columns={"term": str(year)})
49     return df
50
51
52 result_1 = between_region(["socks"],
53                             censor=False,
54                             timeframe="2016-12-14 2017-01-25",
55                             geo="US",
56                             gprop="")
57
58 expected_1 = pd.read_parquet(
59     "tests/test_data/between_region_1.c.parquet")
60
61 print(result_1)
62 print(expected_1)

```

## D.9.2 Get Between-Times Google Search Data

```

1 #!/usr/bin/python
2 from pytrends.request import TrendReq
3 from utils import censor_string
4 import time
5 import pandas as pd
6 import re
7 import pickle as pkl
8
9
10 def in_region(query, region, censor, **kwargs):
11     """ Returns a set of data showing the popularity of a search term over time
12
13     :query: The search term that time-series data is collected for
14     :region: The geographic region to retrieve time-series data for.

```

```

15 :censor: boolean should the search terms be censored?
16 :**kwargs: kwargs to be passed to pytrends.build_payload()
17 :returns: DataFrame giving time-series data for the popularity of a
18 :         search term in a given region
19
20 """
21 try:
22     pytrends = TrendReq(hl='en-US', tz=360)
23     pytrends.build_payload(kw_list=[query], geo=region, **kwargs)
24     df = pytrends.interest_over_time()
25 except:
26     if censor:
27         print(f"Rate error: {censor_string(query)} in {region}")
28     else:
29         print(f"Rate error: {query} in {region}")
30
31     time.sleep(60)
32     return in_region(query, region, censor, **kwargs)
33
34 if df.empty:
35
36     df = pd.DataFrame()
37     df['date'] = pd.period_range(start='2004-01-01',
38                                 end='2021-01-01',
39                                 freq='M').to_timestamp()
40
41     df['n'] = 0
42     df['ispartial'] = pd.Series([True]).bool()
43
44 else:
45
46     df.columns = ["n", "ispartial"]
47     df.index.name = 'date'
48     df.reset_index(inplace=True)
49
50 df["query"] = query
51
52 code = re.findall("\d+", region)
53 if code:
54     df['code'] = code[0]
55 else:
56     code = region
57
58 if censor:
59     df["query"] = df["query"].apply(censor_string)
60
61 return df
62
63 def to_wide(df):
64     """Turns time-series search popularity data into a 'wide' dataframe to be used in scaling
65
66     :df: 'long' dataframe of search data, as from in_region()
67     :returns: 'wide' DataFrame of search data, averaged by year
68
69     """
70     df['year'] = pd.DatetimeIndex(df['date']).year
71     df['year'] = df['year'].apply(str)
72     df = df.groupby(['year', 'code'])["n"].mean()
73     df = df.unstack(level=0)
74     return (df)

```

### D.9.3 Scaling Google Search Data

```

1  #!/usr/bin/env python
2  import numpy as np
3  from utils import mean_nonzero
4  import random
5  import pandas as pd
6
7
8  # Find the ratio of two scores that are horizontally comparable
9  def compare_horizontal(h_matrix, v_matrix, y, x0, x1):
10     if h_matrix[y, x0] == 0 or h_matrix[y, x1] == 0:
11         return None
12     elif v_matrix[y, x0] == 0 or v_matrix[y, x1] == 0:
13         return None
14     else:

```

```

15     present = h_matrix[y, x0]
16     next_val = h_matrix[y, x1]
17     return present / next_val
18
19
20 # Find the ratio of two scores that are vertically comparable
21 def compare_vertical(h_matrix, v_matrix, x, y0, y1):
22     if v_matrix[y0, x] == 0 or v_matrix[y1, x] == 0:
23         return None
24     elif v_matrix[y0, x] == 0 or v_matrix[y1, x] == 0:
25         return None
26     else:
27         present = v_matrix[y0, x]
28         next_val = v_matrix[y1, x]
29         return present / next_val
30
31
32 # Find the ratio of two scores, if possible, using a horizontally comparable
33 # and vertically comparable array
34 def compare(h_matrix, v_matrix, x0, y0, x1, y1):
35     if h_matrix[y0, x0] == 0 or h_matrix[y1, x1] == 0:
36         return (0)
37     elif v_matrix[y0, x0] == 0 or v_matrix[y1, x1] == 0:
38         return (0)
39     else:
40         h_compare_1 = compare_horizontal(h_matrix, v_matrix, y0, x0, x1)
41         v_compare_1 = compare_vertical(h_matrix, v_matrix, x1, y0, y1)
42
43         if h_compare_1 and v_compare_1:
44             first = h_compare_1 * v_compare_1
45         else:
46             first = None
47
48         h_compare_2 = compare_vertical(h_matrix, v_matrix, x0, y0, y1)
49         v_compare_2 = compare_horizontal(h_matrix, v_matrix, y1, x0, x1)
50         if h_compare_2 and v_compare_2:
51             second = h_compare_2 * v_compare_2
52         else:
53             second = None
54
55         if first and second:
56             return (first + second) / 2
57         elif first:
58             return first
59         elif second:
60             return second
61         else:
62             return None
63
64
65 # Find a standardized score for an individual cell given a using the ratio of
66 # the score to other cells, given a horizontally comparable and vertically
67 # comparable array
68 def score_point(h_matrix, v_matrix, x0, y0):
69     ncol, nrow = h_matrix.shape
70
71     if h_matrix[y0, x0] == 0 or v_matrix[y0, x0] == 0:
72         return (0)
73     else:
74         scores = tuple(
75             compare(h_matrix, v_matrix, x0, y0, x1, y1) for y1 in range(ncol)
76             for x1 in range(nrow))
77         return mean_nonzero(scores)
78
79
80 # Find the standardized scores across an entire array, using the ratio of each
81 # score to each other score
82 def recover_scores(h_matrix, v_matrix):
83     assert h_matrix.shape == v_matrix.shape
84     result = np.zeros(h_matrix.shape)
85
86     for y in range(h_matrix.shape[0]):
87         for x in range(h_matrix.shape[1]):
88             result[y, x] = score_point(h_matrix, v_matrix, x, y)
89
90     if np.nanmax(result) == 0:
91         return None

```

```

92     else:
93         result *= 1 / np.nanmax(result)
94     return result
95
96
97 # From a matrix of correctly imputed values and a matrix of horizontally and
98 # vertically comparable values, repair values that could not be imputed in the
99 # first stage because zero values had 'cast a shadow' on on them.
100 def repair_matrix(h_matrix, v_matrix, current):
101     for i in range(500):
102         if i > 1:
103             print(i)
104             current_sum = np.sum(current)
105             current_has_nan = np.isnan(current_sum)
106
107             if current_has_nan:
108                 NAs = np.argwhere(np.isnan(current))
109
110                 not_NA = np.invert(np.isnan(current))
111                 not_zero = current != 0
112                 not_na_zero = np.logical_and(not_NA, not_zero)
113                 not_na_zero_idx = np.argwhere(not_na_zero)
114
115                 for NA_y, NA_x in NAs:
116                     scores = tuple(current[y, x] *
117                                   compare(h_matrix, v_matrix, NA_x, NA_y, x, y)
118                                   for y, x in not_na_zero_idx)
119                     score = mean_nonzero(scores)
120
121                     current[NA_y, NA_x] = score
122             else:
123                 break
124
125     return current
126
127
128 if __name__ == "__main__":
129     arr = np.array([[x + 10 * y for x in range(10)] for y in range(10)])
130
131     rowmod = np.array(tuple(random.random() for _ in range(10)))
132     colmod = np.array(tuple(random.random() for _ in range(10)))
133
134     h_matrix = arr * rowmod[:, np.newaxis]
135     v_matrix = arr * colmod
136
137     n = 10
138     index1 = np.random.choice(arr.size, n, replace=False)
139     index2 = np.random.choice(arr.size, n, replace=False)
140     v_matrix.ravel()[index2] = 0
141     h_matrix.ravel()[index1] = 0
142     np.savetxt("fake_h_mat.csv", h_matrix, delimiter=",")
143     np.savetxt("fake_v_mat.csv", v_matrix, delimiter=",")
144
145     gen1 = recover_scores(h_matrix, v_matrix)
146     np.savetxt("fake_gen1.csv", gen1, delimiter=",")
147     if not gen1 is None:
148
149         repaired = repair_matrix(h_matrix, v_matrix, gen1)
150
151         np.savetxt("fake_repaired.csv", repaired, delimiter=",")

```

## D.10 Unit Tests

### D.10.1 For Python Code

```

1  #!/usr/bin/python
2  import unittest
3  import pandas as pd
4  from between_regions import between_region
5  from utils import connected
6  import warnings
7
8
9  @unittest.skipIf(not connected(), "not connected to the internet")
10 class TestBetweenRegion(unittest.TestCase):
11     def setUp(self):

```

```

12 warnings.simplefilter('ignore')
13
14 def test_between_region_uncensored(self):
15
16     result_1 = between_region(["socks"],
17                               censor=False,
18                               timeframe="2016-12-14 2017-01-25",
19                               geo="US",
20                               gprop="")
21
22     result_2 = between_region(["socks", "shoe", "fish"],
23                               censor=False,
24                               timeframe="2016-12-14 2017-01-25",
25                               geo="US",
26                               gprop="")
27
28     expected_1 = pd.read_parquet(
29         "tests/test_data/between_region_1_uc.parquet")
30     expected_2 = pd.read_parquet(
31         "tests/test_data/between_region_2_uc.parquet")
32
33     self.assertTrue(expected_1.equals(result_1))
34     self.assertTrue(expected_2.equals(result_2))
35
36 def test_between_region_censored(self):
37     result_1 = between_region(["socks"],
38                               censor=True,
39                               timeframe="2016-12-14 2017-01-25",
40                               geo="US",
41                               gprop="")
42
43     result_2 = between_region(["socks", "shoe", "fish"],
44                               censor=True,
45                               timeframe="2016-12-14 2017-01-25",
46                               geo="US",
47                               gprop="")
48
49     expected_1 = pd.read_parquet(
50         "tests/test_data/between_region_1_c.parquet")
51     expected_2 = pd.read_parquet(
52         "tests/test_data/between_region_2_c.parquet")
53
54     self.assertTrue(expected_1.equals(result_1))
55     self.assertTrue(expected_2.equals(result_2))

```

```

1  #!/usr/bin/python
2  import unittest
3  import pandas as pd
4  import warnings
5  from in_region import in_region
6  from utils import connected
7
8
9  @unittest.skipIf(not connected(), "not connected to the internet")
10 class TestBetweenRegion(unittest.TestCase):
11     def setUp(self):
12         warnings.simplefilter('ignore')
13
14     def test_in_region_uncensored(self):
15
16         result_1 = in_region("hello",
17                               "US-AL-630",
18                               False,
19                               timeframe="2016-12-14 2017-01-25")
20
21         result_2 = in_region("France",
22                               "US-GA-522",
23                               False,
24                               timeframe="2016-12-14 2017-01-25")
25
26         expected_1 = pd.read_parquet("tests/test_data/in_region_1_uc.parquet")
27         expected_2 = pd.read_parquet("tests/test_data/in_region_2_uc.parquet")
28
29         self.assertTrue(expected_1.equals(result_1))
30         self.assertTrue(expected_2.equals(result_2))
31
32     def test_in_region_censored(self):

```



```

33     result_1 = in_region("socks",
34                          "US",
35                          True,
36                          timeframe="2016-12-14 2017-01-25",
37                          gprop="")
38
39     expected_1 = pd.read_parquet("tests/test_data/in_region_1_c.parquet")
40
41     self.assertTrue(expected_1.equals(result_1))

```

```

1  #!/usr/bin/python
2  import unittest
3  import numpy as np
4  import random
5  from scaling import compare, compare_horizontal, compare_vertical
6  from scaling import recover_scores, repair_matrix
7
8
9  class TestCompareHorizontal(unittest.TestCase):
10     def test_compare_horizontal(self):
11         h_matrix = np.array([[2, 4, 6], [12, 15, 18], [3.5, 4, 4.5]])
12         v_matrix = np.array([[3, 1, 1], [12, 2.5, 2], [21, 4, 3]])
13
14         self.assertEqual(1 / 3, compare_horizontal(h_matrix, v_matrix, 0, 0,
15                                                     2))
16         self.assertEqual(2 / 3, compare_horizontal(h_matrix, v_matrix, 1, 0,
17                                                     2))
18         self.assertEqual(5 / 4, compare_horizontal(h_matrix, v_matrix, 1, 1,
19                                                     0))
20
21
22  class TestCompareVertical(unittest.TestCase):
23     def test_compare_vertical(self):
24         h_matrix = np.array([[2, 4, 6], [12, 15, 18], [3.5, 4, 4.5]])
25         v_matrix = np.array([[3, 1, 1], [12, 2.5, 2], [21, 4, 3]])
26
27         self.assertEqual(1 / 4, compare_vertical(h_matrix, v_matrix, 0, 0, 1))
28         self.assertEqual(1 / 4, compare_vertical(h_matrix, v_matrix, 1, 0, 2))
29         self.assertEqual(4, compare_vertical(h_matrix, v_matrix, 0, 1, 0))
30
31
32  class TestCompare(unittest.TestCase):
33     def test_compare(self):
34         h_matrix = np.array([[2, 4, 6], [12, 15, 18], [3.5, 4, 4.5]])
35         v_matrix = np.array([[3, 1, 1], [12, 2.5, 2], [21, 4, 3]])
36
37         self.assertEqual(1 / 8, compare(h_matrix, v_matrix, 0, 0, 1, 2))
38         self.assertEqual(1 / 9, compare(h_matrix, v_matrix, 0, 0, 2, 2))
39         self.assertEqual(1 / 4, compare(h_matrix, v_matrix, 1, 0, 1, 2))
40
41
42  class TestRecoverScores(unittest.TestCase):
43     def test_recover_scores(self):
44         for _ in range(100):
45             arr = np.array([[x + 8 * y for x in range(8)] for y in range(8)])
46
47             rowmod = np.array(tuple(random.random() for _ in range(8)))
48             colmod = np.array(tuple(random.random() for _ in range(8)))
49
50             h_matrix = arr * rowmod[:, np.newaxis]
51             v_matrix = arr * colmod
52
53             self.assertTrue(
54                 np.allclose(arr,
55                             recover_scores(h_matrix, v_matrix) * 63))
56
57
58  class TestRepairScores(unittest.TestCase):
59     def test_repair_scores(self):
60         for _ in range(100):
61             arr = np.array([[x + 10 * y for x in range(10)]
62                             for y in range(10)])
63
64             n = 15
65             index = np.random.choice(arr.size, n, replace=False)
66             arr.ravel()[index] = 0
67

```

```

68         rowmod = np.array(tuple(random.random() for _ in range(10)))
69         colmod = np.array(tuple(random.random() for _ in range(10)))
70
71         h_matrix = arr * rowmod[:, np.newaxis]
72         v_matrix = arr * colmod
73
74         gen1 = recover_scores(h_matrix, v_matrix)
75         if not gen1 is None:
76
77             repaired = repair_matrix(h_matrix, v_matrix, gen1)
78
79             repaired_scaled = repaired / np.max(repaired)
80             arr_scaled = arr / np.max(arr)
81
82             self.assertTrue(np.allclose(repaired_scaled, arr_scaled))

```

```

1  #!/usr/bin/python
2  import unittest
3  from utils import itr_split_overlap, censor_string
4
5
6  class TestItrSplitOverlap(unittest.TestCase):
7      def test_itr(self):
8          test_list = ["one", "two", "three", "four", "five"]
9
10         expected_result = [('one', 'two'), ('two', 'three'), ('three', 'four'),
11                             ('four', 'five')]
12         self.assertEqual(expected_result,
13                          list(itr_split_overlap(test_list, 2, 1)))
14
15         expected_result = [('one', 'two', 'three', 'four'),
16                             ('two', 'three', 'four', 'five')]
17         self.assertEqual(expected_result,
18                          list(itr_split_overlap(test_list, 4, 3)))
19
20     def test_exceptions(self):
21         with self.assertRaises(ValueError):
22             test_list = ["one", "two", "three", "four", "five"]
23
24             list(itr_split_overlap(test_list, 2, 3))
25
26
27 class CensorString(unittest.TestCase):
28     def test_censor(self):
29         self.assertEqual("h_y", censor_string("hey"))
30         self.assertEqual("f____r", censor_string("fender"))
31         self.assertEqual("fr", censor_string("fr"))

```

## D.10.2 For R Code

```

1  library(testthat)
2  source("../utils.R")
3
4  test_that("censor string works", {
5      expect_equal(censor_string("hey"), "h_y")
6      expect_equal(censor_string("hy"), "hy")
7      expect_equal(censor_string("watermelon"), "w_____n")
8      expect_equal(censor_string("fantastic"), "f_____c")
9  })
10
11 test_that("censor string vectorized correctly", {
12     expect_equal(censor_string(c("hello", "there")), c("h___o", "t___e"))
13     expect_equal(censor_string(c("watermelon", "paper")), c("w_____n", "p____r"))
14 })
15
16 test_that("years before works", {
17     expect_equal(years_before(c(F, F, F, T, T)), c(-3, -2, -1, 0, 1))
18     expect_equal(years_before(c(F, F, F, F, T)), c(-4, -3, -2, -1, 0))
19     expect_equal(years_before(c(T, T, T, T, T)), c(0, 1, 2, 3, 4))
20 })

```