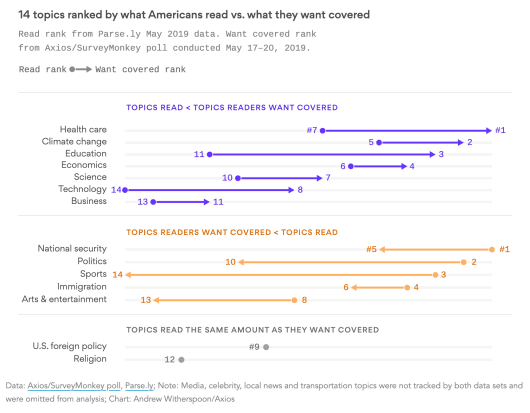
This morning, [@kairyssdal](http://twitter.com/kairyssdal) tweeted out the following graphic from [@axios](http://twitter.com/axios):

Confusing, but interesting.

Data shows we're a nation of news consumption hypocrites – Axios <https://t.co/O0lPSc4OV3>

— Kai Ryssdal ([@kairyssdal](http://twitter.com/kairyssdal)) [June 11, 2019](https://twitter.com/kairyssdal/status/1138405440578961408?ref_src=twsrc%5Etfw)

If you’re doing the right thing and blocking evil social media javascript you can find the Axios story [here](https://www.axios.com/news-consumption-read-topics-56467fe6-81bd-4ae5-9173-cdff9865deda.html) and the graphic below:

[](https://i2.wp.com/rud.is/b/wp-content/uploads/2019/06/axios-01.png?ssl=1)

Its gonna say: the chart is confusing. At the time I had ~5 minutes to spare so I [did a quick makeover](https://twitter.com/hrbrmstr/status/1138413330236878848?s=20) and [a short howto thread](https://twitter.com/hrbrmstr/status/1138416265020747776?s=20). This post is an expansion on said thread and if you’re in the midst of making the decision to consider reading or moving on here’s what we’ll cover:

* Making the choice between scraping or manual data entry
* Quick introduction to [EtherCalc](https://ethercalc.net/)
* Why you might consider using EtherCalc for manual data entry over Excel or Google Sheets
* Pulling data from EtherCalc into R
* Making a slopegraph with the captured data
* Customizing the slopegraph with the captured data to tell one or more stories

Read on if any or all of that is captures your interest.

**To scrape or not to scrape**

Even if I wanted to scrape the site, Axios [makes it pretty clear they are kinda not very nice people](https://www.axios.com/legal) since — while it doesn’t mention scraping — that ToS link does indicate that:

(a) you will not license, sell, rent, lease, transfer, assign, distribute, host, or otherwise commercially exploit the Site or any content displayed on the Site; (b) you will not modify, make derivative works of, disassemble, reverse compile or reverse engineer any part of the Site; (c) you will not access the Site in order to build a similar or competitive website, product, or service; and (d) except as expressly stated herein, no part of the Site may be copied, reproduced, distributed, republished, downloaded, displayed, posted or transmitted in any form or by any means. Unless otherwise indicated, any future release, update, or addition to the Site’s functionality will be subject to these Terms. All copyright and other proprietary notices on the Site (or on any content displayed on the Site) must be retained on all copies thereof.

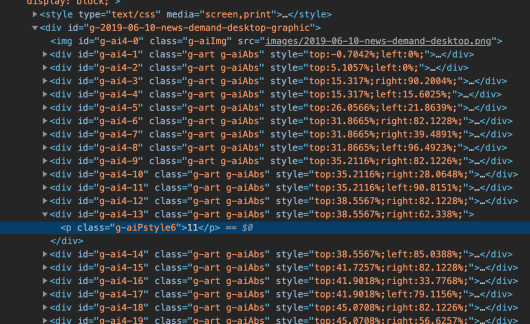
(OH NO I COPIED THAT FROM THEIR SITE SO I AM ALREADY VIOLATING THEIR [unenforceable] TOS!)

There’s this thing called [“Fair Use”](https://fairuse.stanford.edu/overview/fair-use/what-is-fair-use/) and this makeover I’m doing is 100% covered under that. The Axios ToS and the ToS of many other sites try to prohibit such fair use and they generally lose those battles in court. I have and will be citing them as sources throughout this post and the post itself falls under “commentary and criticism”. Unlike many unethical scrapers who are just scavenging data they did no work to generate and whose work will not serve the better interest of the general community, this post is fully dedicated to sharing and education.

In reality, Axios likely has such draconian ToS due to all the horrible unethical scrapers who just want free, un-cited news content.

*Anyway*…

Even if I could scrape they don’t embed a javascript data object nor do they load an XHR JSON data blob to make the graphic. They use an idiom of loading a base image then perform annotation via markup:

[](https://i0.wp.com/rud.is/b/wp-content/uploads/2019/06/axios-graphic-1.png?ssl=1)

making it not worth taking the time to scrape.

That means data entry. 

**Using EtherCalc for ~~fun and profit~~ data entry**

I dislike Microsoft Excel (even the modern versions of it) because it is overkill for data entry. I also dislike performing data entry in Google Sheets since that means I need to be cloud-connected. So, for small, local data entry needs I turn to [EtherCalc](https://ethercalc.net/). No internet access is required, nor is there a bloated app to run.

EtherCalc is a *multiuser* Google Sheets-like browser-based spreadsheet powered by javascript (both in-browser and the back-end). You can install it with:

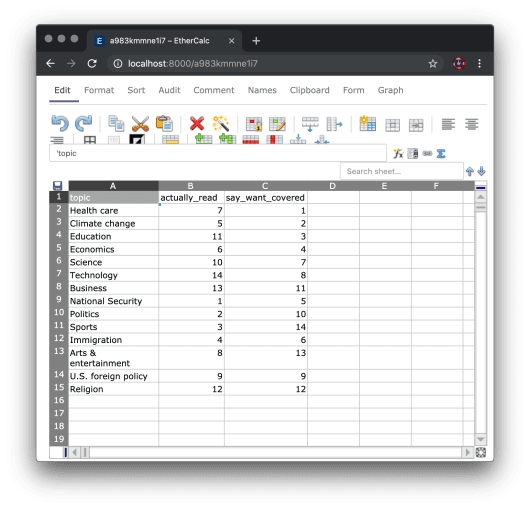
$ npm install -g ethercalc

which assumes you have a working [Node.js](https://nodejs.org/en/) setup along with [npm](https://www.npmjs.com/).

When you run:

$ ethercalc

you are given a URL to hit with your browser. Below is what that looks like with my data entry already complete:

[](https://i2.wp.com/rud.is/b/wp-content/uploads/2019/06/ethercalc-1.png?ssl=1)

It can use Redis or a local filesystem as a persistence layer and does support multiple folks editing the same document at the same time.

At this point I could just save it out manually to a CSV file and read it in the old-fashioned way, but EtherCalc has [an API](https://github.com/audreyt/ethercalc/blob/master/API.md)! So we can grab the data using {httr} calls, like this:

library(hrbrthemes)

library(tidyverse)

httr::GET(

url = "http://localhost:8000/a983kmmne1i7.csv"

) -> res

(xdf <- httr::content(res))

## # A tibble: 14 x 3

## topic actually\_read say\_want\_covered

##

## 1 Health care 7 1

## 2 Climate change 5 2

## 3 Education 11 3

## 4 Economics 6 4

## 5 Science 10 7

## 6 Technology 14 8

## 7 Business 13 11

## 8 National Security 1 5

## 9 Politics 2 10

## 10 Sports 3 14

## 11 Immigration 4 6

## 12 Arts & entertainment 8 13

## 13 U.S. foreign policy 9 9

## 14 Religion 12 12

where a983kmmne1i7 is the active document identifer.

Now that we have the data, it’s time to start the makeover.

**Stage 1: A basic slopegraph**

(If you need a primer on slopegraphs, definitely check out [this resource](http://www.visualisingdata.com/2013/12/in-praise-of-slopegraphs/) by [@visualisingdata](http://twitter.com/visualisingdata).)

We need to make a decision as to what’s going where on the slopegraph. I’m choosing to put what respondents *actually read* on the left and then what they *say they want covered* on the right. Regardless of order, we need to do bit of data wrangling to take a first stab at the chart:

ggplot() +

# draw the slope lines

geom\_segment(

data = xdf,

aes(

x = "Actually read", y = actually\_read,

xend = "Say they\nwant covered", yend = say\_want\_covered

)

) +

# left and right vertical bars

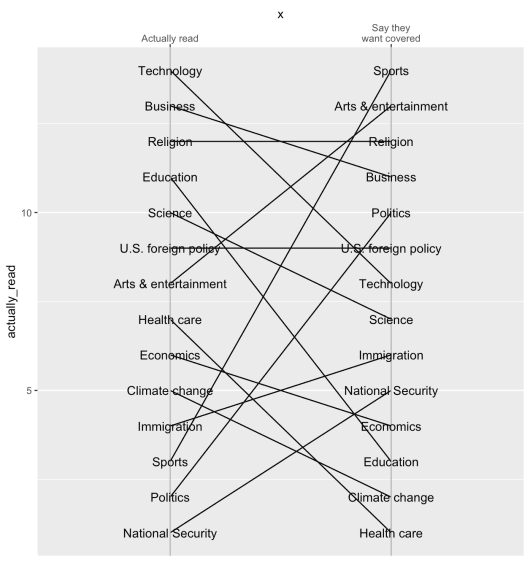
geom\_vline(aes(xintercept = c(1, 2)), color = "#b2b2b2") +

# left and right category text

geom\_text(data = xdf, aes("Actually read", actually\_read, label = topic)) +

geom\_text(data = xdf, aes("Say they\nwant covered", say\_want\_covered, label = topic)) +

scale\_x\_discrete(position = "top")

[](https://i0.wp.com/rud.is/b/wp-content/uploads/2019/06/slope-01a.png?ssl=1)

That chart isn’t winning any (good) awards any time soon. Apart from the non-aligned category labels, the categories aren’t in traditional order (rank “#1” being at the top on the left), plus we definitely need more information on the chart (title, subtitle, caption, etc.). We’ll reorder the labels and tweak some of the aesthetic problems away and switch the theme:

xdf <- mutate(xdf, dir = factor(sign(actually\_read - say\_want\_covered))) # get the category order right

xdf <- mutate(xdf, actually\_read = -actually\_read, say\_want\_covered = -say\_want\_covered) # reverse the Y axis

ggplot() +

geom\_segment(

data = xdf,

aes(

"Actually read", actually\_read,

xend = "Say they\nwant covered", yend = say\_want\_covered

),

size = 0.25, color = "#b2b2b2"

) +

geom\_vline(aes(xintercept = c(1, 2)), color = "#b2b2b2") +

geom\_text(

data = xdf,

aes("Actually read", actually\_read, label = topic),

family = font\_rc, size = 4, hjust = 1, nudge\_x = -0.01

) +

geom\_text(

data = xdf,

aes("Say they\nwant covered", say\_want\_covered, label = topic),

family = font\_rc, size = 4, hjust = 0, nudge\_x = 0.01

) +

scale\_x\_discrete(position = "top") +

labs(

x = NULL, y = NULL,

title = "14 Topics Ranked by What Americans Read vs Want Covered",

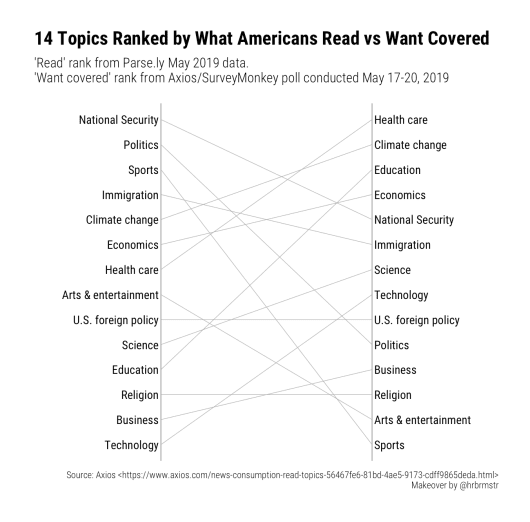
subtitle = "'Read' rank from Parse.ly May 2019 data.\n'Want covered' rank from Axios/SurveyMonkey poll conducted May 17-20, 2019",

caption = "Source: Axios \nMakeover by [@hrbrmstr](http://twitter.com/hrbrmstr)"

) +

theme\_ipsum\_rc(grid="") +

theme(axis.text = element\_blank())

[](https://i1.wp.com/rud.is/b/wp-content/uploads/2019/06/slope-02a.png?ssl=1)

That looks much better and I stopped there due to time constraints for the initial thread. However, the slope lines tend to be fairly hard to follow and we really should be telling a story with them. But what story do we want to focus on ?

**Story time**

One aesthetic element we’ll want to immediately modify regardless of story is the line color. We can use the dir column for this:

ggplot() +

geom\_segment(

data = xdf,

aes(

"Actually read", actually\_read,

xend = "Say they\nwant covered", yend = say\_want\_covered,

color = dir, size = dir

)

) +

geom\_vline(aes(xintercept = c(1, 2)), color = "#b2b2b2") +

geom\_text(

data = xdf,

aes("Actually read", actually\_read, label = topic),

family = font\_rc, size = 4, hjust = 1, nudge\_x = -0.01, lineheight = 0.875

) +

geom\_text(

data = xdf,

aes("Say they\nwant covered", say\_want\_covered, label = topic),

family = font\_rc, size = 4, hjust = 0, nudge\_x = 0.01, lineheight = 0.875

) +

scale\_x\_discrete(position = "top") +

scale\_size\_manual(

values = c(

`-1` = 0.2,

`0` = 0.2,

`1` = 0.2

),

) +

scale\_color\_manual(

name = NULL,

values = c(

`-1` = ft\_cols$red,

`0` = "#2b2b2b",

`1` = ft\_cols$blue

),

labels = c(

`-1` = "Topics Readers Want Covered < Topics Read",

`0` = "Topics Read The Same Amount As They Want Covered",

`1` = "Topics Read < Topics Readers Want Covered"

)

) +

guides(

size = FALSE

) +

labs(

x = NULL, y = NULL,

title = "14 Topics Ranked by What Americans Read vs Want Covered",

subtitle = "'Read' rank from Parse.ly May 2019 data.\n'Want covered' rank from Axios/SurveyMonkey poll conducted May 17-20, 2019",

caption = "Source: Axios \nMakeover by [@hrbrmstr](http://twitter.com/hrbrmstr)"

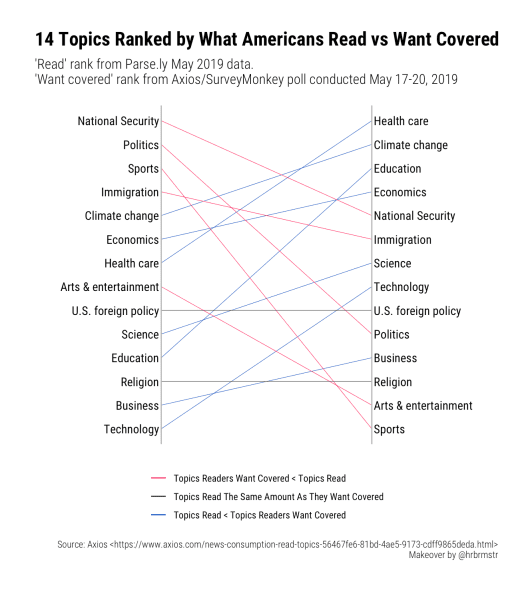
) +

theme\_ipsum\_rc(grid="") +

theme(axis.text = element\_blank()) +

theme(legend.position = "bottom") +

theme(legend.direction = "vertical")

[](https://i2.wp.com/rud.is/b/wp-content/uploads/2019/06/slope-03.png?ssl=1)

It’s still somewhat hard to pick out stories and the legend may be useful but it’s not ideal. Let’s highlight the different slope types with color, annotate them directly, and see what emerges:

library(hrbrthemes)

library(tidyverse)

httr::GET(

url = "http://localhost:8000/a983kmmne1i7.csv"

) -> res

(xdf <- httr::content(res))

xdf <- mutate(xdf, dir = factor(sign(actually\_read - say\_want\_covered)))

xdf <- mutate(xdf, actually\_read = -actually\_read, say\_want\_covered = -say\_want\_covered)

arw <- arrow(length = unit(5, "pt"), type = "closed")

# x = c(1.2, 1.8, 1.9),

# y = -c(1, 13, 14),

# xend = c(1.05, 1.7, 1.6),

# yend = -c(1.125, 13, 14)

# ),

# aes(x, y , xend=xend, yend=yend),

ggplot() +

geom\_segment(

data = xdf,

aes(

"Actually read", actually\_read,

xend = "Say they\nwant covered", yend = say\_want\_covered,

color = dir, size = dir

), show.legend = FALSE

) +

geom\_vline(aes(xintercept = c(1, 2)), color = "#b2b2b2") +

geom\_text(

data = xdf,

aes("Actually read", actually\_read, label = topic),

family = font\_rc, size = 4, hjust = 1, nudge\_x = -0.01, lineheight = 0.875

) +

geom\_text(

data = xdf,

aes("Say they\nwant covered", say\_want\_covered, label = topic),

family = font\_rc, size = 4, hjust = 0, nudge\_x = 0.01, lineheight = 0.875

) +

geom\_curve(

data = data.frame(),

aes(x = 1.2, y = -1, xend = 1.05, yend = -1.125),

color = ft\_cols$red, arrow = arw

) +

geom\_segment(

data = data.frame(), aes(x = 1.6, xend = 1.6, yend = -12.1, y = -12.9),

color = "#2b2b2b", arrow = arw

) +

geom\_curve(

data = data.frame(), aes(x = 1.2, y = -14.1, xend = 1.1, yend = -13.6),

curvature = -0.5, color = ft\_cols$blue, arrow = arw

) +

geom\_text(

data = data.frame(

x = c(1.15, 1.6, 1.2),

y = -c(1.2, 13, 14),

hjust = c(0, 0.5, 0),

vjust = c(0.5, 1, 0.5),

lab = c(

"Topics Readers Want Covered < Topics Read",

"Topics Read The Same Amount\nAs They Want Covered",

"Topics Read < Topics Readers Want Covered"

),

stringsAsFactors = FALSE

),

aes(x, y, hjust = hjust, vjust = vjust, label = lab),

family = font\_rc, size = 2.5, lineheight = 0.875

) +

scale\_x\_discrete(position = "top") +

scale\_size\_manual(

values = c(

`-1` = 0.75,

`0` = 0.2,

`1` = 0.2

)

) +

scale\_color\_manual(

name = NULL,

values = c(

`-1` = ft\_cols$red,

`0` = "#2b2b2b",

`1` = ft\_cols$blue

)

) +

labs(

x = NULL, y = NULL,

title = "14 Topics Ranked by What Americans Read vs Want Covered",

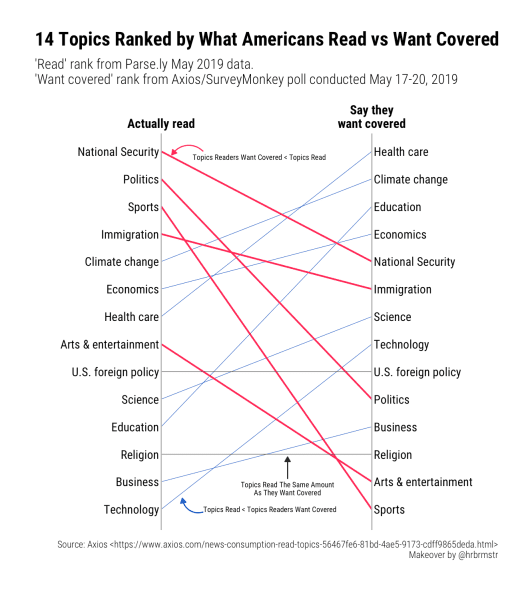
subtitle = "'Read' rank from Parse.ly May 2019 data.\n'Want covered' rank from Axios/SurveyMonkey poll conducted May 17-20, 2019",

caption = "Source: Axios \nMakeover by [@hrbrmstr](http://twitter.com/hrbrmstr)"

) +

theme(axis.text.x = element\_text(size = 12, face = "bold", color = "black")) +

theme(axis.text.y = element\_blank())

[](https://i0.wp.com/rud.is/b/wp-content/uploads/2019/06/slope-04.png?ssl=1)

This first story indicates a potential social desirability bias in the respondents in that they claim to care more about health care, climate change, and education but really care more about more frivolous things (sports), base things (politics), and things they have almost no control over (national security).

Let’s switch the focus (only showing the modified aesthetic to avoid a code DoS):

scale\_size\_manual(

values = c(

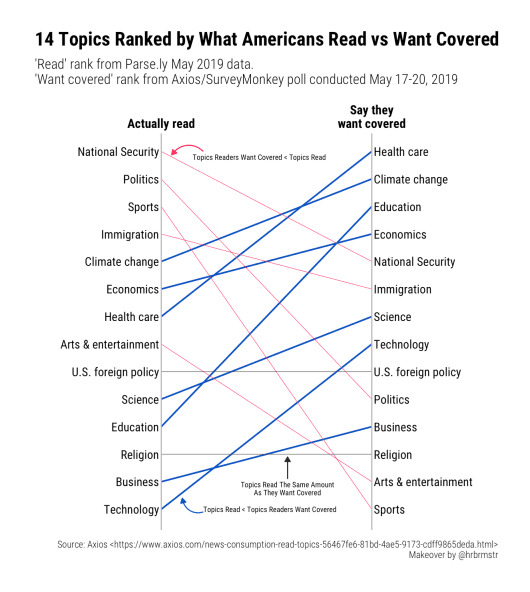
`-1` = 0.2,

`0` = 0.2,

`1` = 0.75

)

) +

[](https://i1.wp.com/rud.is/b/wp-content/uploads/2019/06/slope-05.png?ssl=1)

Now we get to see just how far down on the priority list some of the “desired coverage” topics really sit. At least Health care is not at the bottom, but given how much technology controls our lives it’s a bit disconcerting to see that at the bottom.

What about the categories that did not differ in rank:

scale\_size\_manual(

values = c(

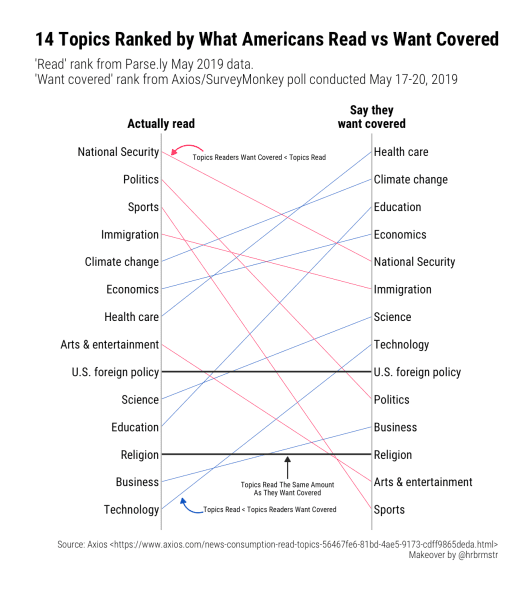
`-1` = 0.2,

`0` = 0.75,

`1` = 0.2

)

) +

[](https://i2.wp.com/rud.is/b/wp-content/uploads/2019/06/slope-06.png?ssl=1)

You’re guess is as good as mine why folks rated these the same (assuming the surveys had similar language).

**FIN**

Now that you’ve got the data (oh, right, I forgot to do that):

structure(list(topic = c("Health care", "Climate change", "Education",

"Economics", "Science", "Technology", "Business", "National Security",

"Politics", "Sports", "Immigration", "Arts & entertainment",

"U.S. foreign policy", "Religion"), actually\_read = c(7, 5, 11,

6, 10, 14, 13, 1, 2, 3, 4, 8, 9, 12), say\_want\_covered = c(1,

2, 3, 4, 7, 8, 11, 5, 10, 14, 6, 13, 9, 12)), class = c("spec\_tbl\_df",

"tbl\_df", "tbl", "data.frame"), row.names = c(NA, -14L), spec = structure(list(

cols = list(topic = structure(list(), class = c("collector\_character",

"collector")), actually\_read = structure(list(), class = c("collector\_double",

"collector")), say\_want\_covered = structure(list(), class = c("collector\_double",

"collector"))), default = structure(list(), class = c("collector\_guess",

"collector")), skip = 1), class = "col\_spec"))

and some alternate views, perhaps you have an even better way to look at it. Drop a note in the comments with any of your creations or suggestions for improvement for the final versions shown here.