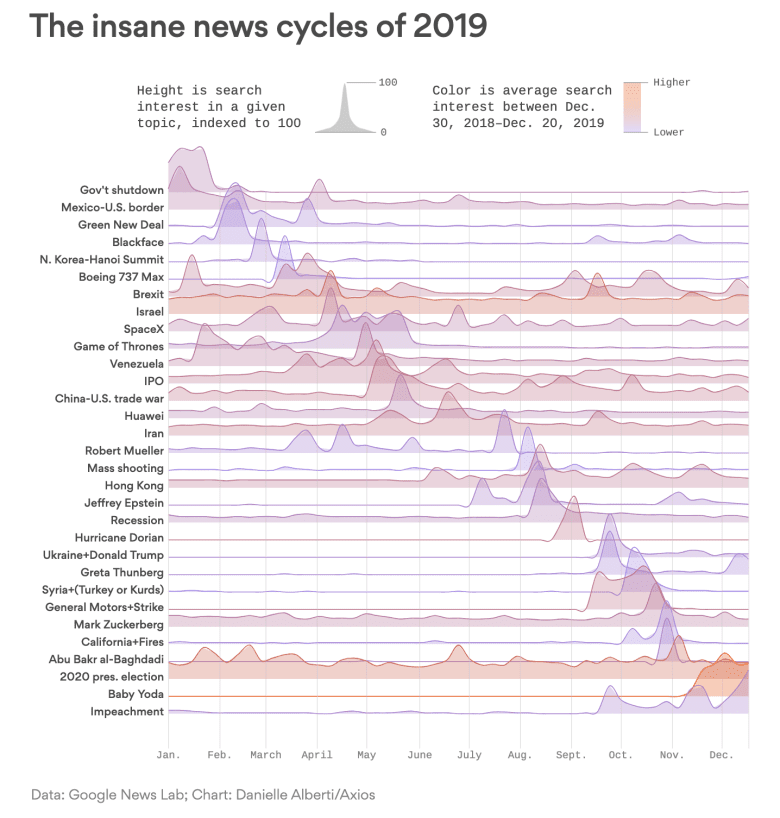
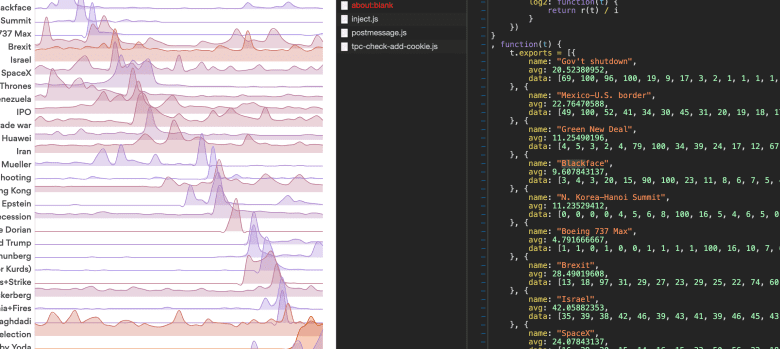
We woke up to Axios’ [“1 Big Thing”](https://www.axios.com/insane-news-cycle-attention-2019-google-interest-f89e7bc8-399d-4681-b479-00e8b6a4ccbc.html) ridgeline chart showing the crazy that was the 2019 news cycle:



and, I decided to reproduce it in {ggplot2}.

**Getting The Data**

First, I had to find the data. The Axios chart is interactive, so I assumed the visualization was built on-load. It was, but the data was embedded in a javascript file vs loaded as JSON via an [XHR request](https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest):



which was easy enough to turn into [JSON](https://rud.is/dl/2019-axios-news.json) anyone can use.

library(ggalt)

library(hrbrthemes) # hrbrmstr/hrbrthemes

library(tidyverse)

jsonlite::fromJSON("<https://rud.is/dl/2019-axios-news.json>") %>%

as\_tibble() -> xdf

xdf

## # A tibble: 31 x 3

## name avg data

##

## 1 Gov't shutdown 20.5

## 2 Mexico-U.S. border 22.8

## 3 Green New Deal 11.3

## 4 Blackface 9.61

## 5 N. Korea-Hanoi Summit 11.2

## 6 Boeing 737 Max 4.79

## 7 Brexit 28.5

## 8 Israel 42.1

## 9 SpaceX 24.1

## 10 Game of Thrones 16.8

## # … with 21 more rows

This is pretty tidy already, but we’ll need to expand the data column and give each week an index:

unnest(xdf, data) %>%

group\_by(name) %>%

mutate(idx = 1:n()) %>%

ungroup() %>%

mutate(name = fct\_inorder(name)) -> xdf # making a factor foe strip/panel ordering

xdf

## # A tibble: 1,581 x 4

## name avg data idx

##

## 1 Gov't shutdown 20.5 69 1

## 2 Gov't shutdown 20.5 100 2

## 3 Gov't shutdown 20.5 96 3

## 4 Gov't shutdown 20.5 100 4

## 5 Gov't shutdown 20.5 19 5

## 6 Gov't shutdown 20.5 9 6

## 7 Gov't shutdown 20.5 17 7

## 8 Gov't shutdown 20.5 3 8

## 9 Gov't shutdown 20.5 2 9

## 10 Gov't shutdown 20.5 1 10

## # … with 1,571 more rows

We’ll take this opportunity to find the first week of each month (via rle()) so we can have decent axis labels:

# get index placement for each month axis label

sprintf("2019-%02s-1", 1:51) %>%

as.Date(format = "%Y-%W-%w") %>%

format("%b") %>%

rle() -> mons

mons

## Run Length Encoding

## lengths: int [1:12] 4 4 4 5 4 4 5 4 5 4 ...

## values : chr [1:12] "Jan" "Feb" "Mar" "Apr" "May" "Jun" "Jul" "Aug" "Sep" "Oct" ...

month\_idx <- cumsum(mons$lengths)-3

month\_idx

## [1] 1 5 9 14 18 22 27 31 36 40 44 48

We’ve got all we need to make a {ggplot2} version of the chart. Here’s the plan:

* use geom\_area() and map colour and fill to avg (like Axios did), using an medium alpha value so we can still see below the overlapped areas
* also use an xspline() stat with geom\_area() so we get smooth lines vs pointy ones
* use geom\_hline() vs an axis line so we can map a colour aesthetic to avg as well
* make a custom x-axis scale so we can place the labels we just made
* expand the y-axis upper limit to avoid cutting off any part of the geoms
* use the inferno viridis palette, but not the extremes of it
* make facets/panels on the name, positioning the labels on the left
* finally, tweak strip positioning so we get overlapped charts

ggplot(xdf, aes(idx, data)) +

geom\_area(alpha = 1/2, stat = "xspline", aes(fill = avg, colour = avg)) +

geom\_hline(

data = distinct(xdf, name, avg),

aes(yintercept = 0, colour = avg), size = 0.5

) +

scale\_x\_continuous(

expand = c(0,0.125), limits = c(1, 51),

breaks = month\_idx, labels = month.abb

) +

scale\_y\_continuous(expand = c(0,0), limits = c(0, 105)) +

scale\_colour\_viridis\_c(option = "inferno", direction = -1, begin = 0.1, end = 0.9) +

scale\_fill\_viridis\_c(option = "inferno", direction = -1, begin = 0.1, end = 0.9) +

facet\_wrap(~name, ncol = 1, strip.position = "left", dir = "h") +

labs(

x = NULL, y = NULL, fill = NULL, colour = NULL,

title = "1 big thing: The insane news cycles of 2019",

subtitle = "Height is search interest in a given topic, indexed to 100.\nColor is average search interest between Dec. 30, 2018–Dec. 20, 2019",

caption = "Source: Axios \nData: Google News Lab. Orig. Chart: Danielle Alberti/Axios"

) +

theme\_ipsum\_es(grid="X", axis = "") +

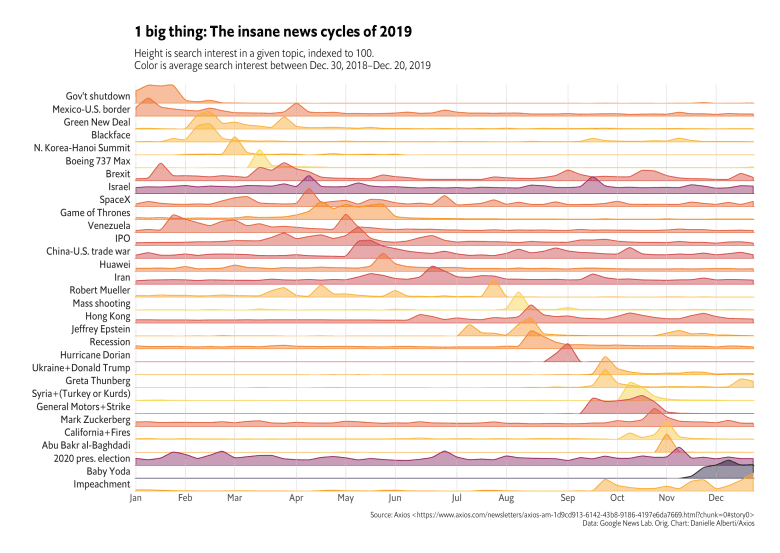
theme(strip.text.y = element\_text(angle = 180, hjust = 1, vjust = 0)) +

theme(panel.spacing.y = unit(-0.5, "lines")) +

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

theme(legend.position = "none")

To produce this finished product:



**FIN**

The chart could be tweaked a bit more to get even closer to the Axios finished product.

Intrepid readers can also try to use {plotly} to make an interactive version.

Somehow, I get the feeling 2020 will have an even more frenetic news cycle.