Today’s RSS feeds picked up [this article](https://theconversation.com/the-epa-has-backed-off-enforcement-under-trump-here-are-the-numbers-108640) by Marianne Sullivan, Chris Sellers, Leif Fredrickson, and Sarah Lamdanon on the woeful state of enforcement actions by the U.S. Environmental Protection Agency (EPA). While there has definitely been overreach by the EPA in the past the vast majority of its regulatory corpus is quite sane and has made Americans safer and healthier as a result. What’s happened to an EPA left in the hands of evil (yep, “evil”) in the past two years is beyond lamentable and we likely have two more years of lamenting ahead of us (unless you actually like your water with a coal ash chaser).

The authors of the article made this chart to show the stark contrast between 2017 and 2018 when it comes to regulatory actions for eight acts:

* Clean Air Act (CAA)
* Clean Water Act (CWA)
* Emergency Planning and Community Right to Know Act (EPCRA)
* Federal Insecticide, Fungicide, and Rodenticide Act (FIFRA)
* Resource Conservation and Recovery Act (RCRA)
* Safe Drinking Water Act (SDWA)
* Toxic Substances Control Act (TSCA)
* Comprehensive Environmental Response, Compensation, and Liability Act (CERCLA)

They made this arrow chart (via [Datawrapper](https://www.datawrapper.de/)):

For some reason, that chart sparked a *“I really need to make that in R”* moment, and thus begat this post.

I’ve got a geom for [dumbbell charts](https://gitlab.com/hrbrmstr/ggalt/blob/master/R/geom_dumbbell.R) but that’s not going to work for this arrow chart since I really wanted to (mostly) reproduce it the way it was. Here’s my go at it.

**Data First**

Datawrapper embeds have a handy “Get the data” link in them but it’s not a link to a file. It’s a javascript-generated data: href so you either need to click on the link and download it *or* be hard-headed like I am go the way of pain and scrape it (reproducibility FTW). Let’s get packages and data gathering code out of the way. I’ll exposit a bit more about said data gathering after the code block:

library(stringi)

library(rvest)

library(hrbrthemes) # git[la|hu]b / hrbrmstr / hrbrthemes

library(tidyverse)

article <- read\_html("https://theconversation.com/the-epa-has-backed-off-enforcement-under-trump-here-are-the-numbers-108640")

html\_node(article, "iframe#psm7n") %>% # find the iframe

html\_attr("src") %>% # get iframe URL

read\_html() %>% # read it in

html\_node(xpath=".//script[contains(., 'data: ')]") %>% # find the javascript section with the data

html\_text() %>% # get that section

stri\_split\_lines() %>% # split into lines so we can target the actual data element

unlist() %>%

keep(stri\_detect\_fixed, 'data: "Fiscal') %>% # just get the data line

stri\_trim\_both() %>% # prep it for extraction

stri\_replace\_first\_fixed('data: "', "") %>%

stri\_replace\_last\_fixed('"', "") %>%

stri\_replace\_all\_fixed("\\n", "\n") %>% # make lines lines

stri\_split\_lines() %>%

unlist() %>%

stri\_split\_fixed("\\t") %>% # we now have a list of vectors

map\_dfc(~set\_names(list(.x[2:length(.x)]), .x[1])) %>% # first element of each vector is colname

type\_convert(col\_types = "cddn") %>% # get real types

set\_names(c("act", "y2018", "y2017", "pct")) -> psm

psm

## # A tibble: 8 x 4

## act y2018 y2017 pct

##

## 1 CAA 199 405 -51

## 2 CERCLA 147 194 -24

## 3 CWA 320 565 -43

## 4 EPCRA 56 107 -48

## 5 FIFRA 363 910 -60

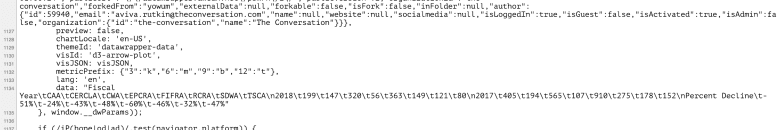
## 6 RCRA 149 275 -46

## 7 SDWA 121 178 -32

## 8 TSCA 80 152 -47

Inside the main article URL content there’s an iframe load:

We grab the contents of that iframe link (https://datawrapper.dwcdn.net/psm7n/2/) which has a data: line way down towards the bottom of one of the last javascript blocks:

[](https://rud.is/b/2019/01/03/data-scraping-chart-reproduction-arrows-of-environmental-destruction/data-line/)

That ugly line gets transformed into a link that will download as a normal CSV file, but we have to do the above wrangling on it before we can get it into a format we can work with.

Now, we can make the chart.

**Chart Time!**

Let’s get the Y axis in the right order:

psm %>%

arrange(desc(y2017)) %>%

mutate(act = factor(act, levels = rev(act))) -> psm

Next, we setup X axis breaks and also get the max value for some positioning calculations (so we don’t hardcode values):

# setup x axis breaks and max value for label position computation

x\_breaks <- pretty(c(psm$y2018, psm$y2017))

max\_val <- max(x\_breaks)

I have two minor nitpicks about the original chart (and changes to them as a result). First, I really don’t like the Y axis gridlines but I do believe we need something to help the eye move horizontally and associate each label to its respective geom. Instead of gridlines I opt for a diminutive dotted line from 0 to the first (min) value.

The second nitpick is that — while the chart has the act information in the caption area — the caption is in alpha order vs the order the act acronyms appear in the data. If it was an alpha bullet list I might not complain, but I chose to modify the order to fit the chart, which we build dynamically with the help of this vector:

# act info for caption

c(

"CAA" = "Clean Air Act (CAA)",

"CWA" = "Clean Water Act (CWA)",

"EPCRA" = "Emergency Planning and Community Right to Know Act (EPCRA)",

"FIFRA" = "Federal Insecticide, Fungicide, and Rodenticide Act (FIFRA)",

"RCRA" = "Resource Conservation and Recovery Act (RCRA)",

"SDWA" = "Safe Drinking Water Act (SDWA)",

"TSCA" = "Toxic Substances Control Act (TSCA)",

"CERCLA" = "Comprehensive Environmental Response, Compensation, and Liability Act (CERCLA)"

) -> acts

w125 <- scales::wrap\_format(125) # help us word wrap at ~125 chars

# order the vector and turn it into wrapped lines

act\_info <- w125(paste0(unname(acts[as.character(psm$act)]), collapse = "; "))

Now, we can generate the geoms. It looks like alot of code, but I like to use newlines to help structure ggplot2 calls. I still miss my old gg <- gg + idiom but RStudio makes it way too easy to execute the whole expression with just the use of + so I’ve succumbed to their behaviour modification. To break it down w/o code, we essentially need:

* the arrows for each act
* the 2017 and 2018 direct label values for each act
* the 2017 and 2018 top “titles”
* segments for ^^
* title, subtitle and caption(s)

We use percent-maths to position labels and other objects so the code can be re-used for other arrow plots (hardcoding to the data values is likely fine, but you’ll end up tweaking the numbers more and wasting ~2-5m per new chart).

ggplot(psm) +

# dots from 0 to minval

geom\_segment(

aes(0, act, xend = y2018, yend = act),

linetype = "dotted", color = "#b2b2b2", size = 0.33

) +

# minval label

geom\_label(

aes(y2018, act, label = y2018),

label.size = 0, hjust = 1, size = 3.5, family = font\_rc

) +

# maxval label

geom\_label(

aes(y2017 + (0.0015 \* y2017), act, label = y2017),

label.size = 0, hjust = 0, size = 3.5, family = font\_rc

) +

# the measure line+arrow

geom\_segment(

aes(y2018, act, xend = y2017, yend = act),

color = "#4a90e2", size = 0.75, # I pulled the color value from the original chart

arrow = arrow(ends = "first", length = unit(5, "pt"))

) +

# top of chart year (min)

geom\_label(

data = head(psm, 1),

aes(y2018, 9, label = "2018"),

hjust = 0, vjust = 1, label.size = 0, size = 3.75, family = font\_rc, color = ft\_cols$slate

) +

# top of chart year (max)

geom\_label(

data = head(psm, 1),

aes(y2017, 9, label = "2017"),

hjust = 1, vjust = 1, label.size = 0, size = 3.75, family = font\_rc, color = ft\_cols$slate

) +

# bar from top of chart year label to first minval measure

geom\_segment(

data = head(psm, 1),

aes(

y2018 + (0.005 \* max\_val), 8.5,

xend = y2018 + (0.005 \* max\_val), yend = 8.25

),

size = 0.25

) +

# bar from top of chart year label to first maxval measure

geom\_segment(

data = head(psm, 1),

aes(

y2017 - (0.005 \* max\_val), 8.5,

xend = y2017 - (0.005 \* max\_val), yend = 8.25

),

size = 0.25

) +

# fix x axis scale and place breaks

scale\_x\_comma(limits = c(0, max\_val), breaks = seq(0, max\_val, 200)) +

# make room for top "titles"

scale\_y\_discrete(expand = c(0, 1)) +

labs(

y = NULL,

title = "Decline by statute",

subtitle = "The number of civil cases the EPA brought to conclusion has dropped across a number of federal statutes,\nincluding the Clean Air Act (CAA) and others.",

x = act\_info,

caption = "Original Chart/Data: The Conversation, CC-BY-ND;; Source: Environmental Data & Government Initiative "

) +

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

theme(axis.text.x = element\_text(color = ft\_cols$slate)) +

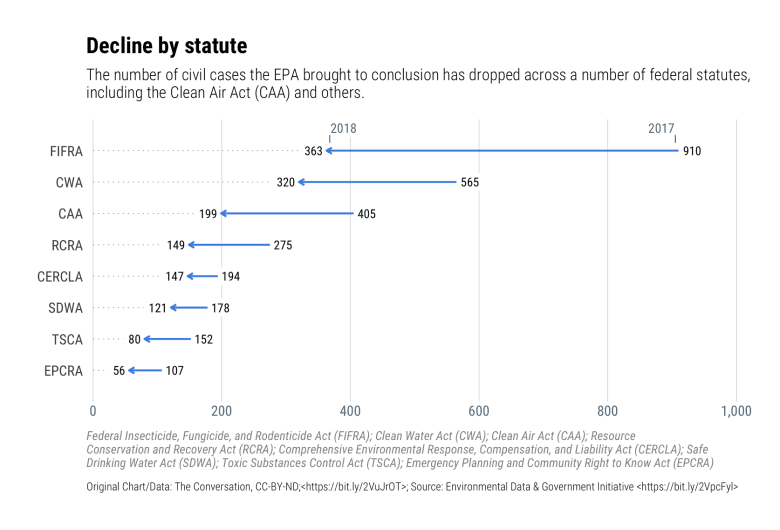
theme(axis.title.x = element\_text(

hjust = 0, size = 10, face = "italic", color = ft\_cols$gray, margin = margin(t = 10)

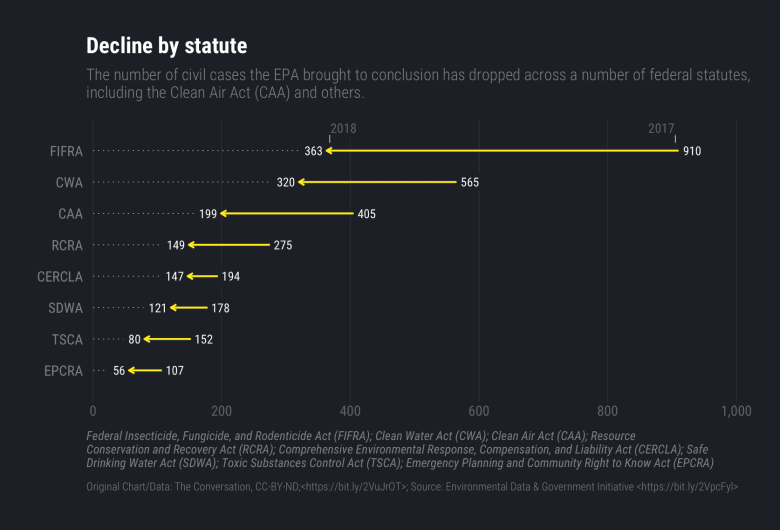
)) +

theme(plot.caption = element\_text(hjust = 0))

Here’s the result:

[](https://rud.is/b/2019/01/03/data-scraping-chart-reproduction-arrows-of-environmental-destruction/chart-makeover/)

(it even looks ok in “batman” mode):

[](https://rud.is/b/2019/01/03/data-scraping-chart-reproduction-arrows-of-environmental-destruction/plot_zoom_png-13/)

**FIN**

With Microsoft owning GitHub I’m not using gists anymore and the GitLab “snippets” equivalent is just too dog-slow to use, so starting in 2019 I’m self-hosing contiguous R example code used in the blog posts. For the moment, that means links to plain R files but I may just setup gitea for them sometime before the end of Q1. You can find a contiguous, commented version of the above code in [here](https://rud.is/r-code/2018-01-03-convo-epa-chart.R).