There’s a *yuge* chance you’re reading this post (at least initially) on R-Bloggers right now .It’s a central “watering hole” for R folks and is read by many (IIRC over 20,000 Feedly users have it in their OPML).

I’m *addicted* to [Feedly](https://feedly.com/) and waited *years* for their API. There will eventually be a package for it (go for it if you want to get’er done before me since I won’t have time to do it justice for a while). As just parenthetically noted, I’ve started work on one and have scaffolded *just enough* to give R folks a present: almost 5 years of R-Bloggers data — posts, engagement rates, authors, etc). *But*, you’ll have to put up with some expository, first.

**Digging In**

We’ll need some packages to help this expository and extraction. Plus, you’ll need to go to <https://developer.feedly.com/> to get your developer token (NOTE: this requires a “Pro” account *or* a regular account and you manually doing the OAuth dance to get an access token; any final “Feedly package” by myself or others will likely use OAuth) and store it in your ~/.Renviron in FEEDLY\_ACCESS\_TOKEN.

I’ve sliced and diced bits from the (non-published) fledgling package to give a peek behind the API covers. There’s *plenty* of exposition in the following code block comment header to describe what it does:

#' Simplifying some example package setup for this non-pkg example

.pkgenv <- new.env(parent=emptyenv())

.pkgenv$token <- Sys.getenv("FEEDLY\_ACCESS\_TOKEN")

#' In reality, this is more complex since the non-toy example has to

#' refresh tokens when they expire.

.feedly\_token <- function() {

return(.pkgenv$token)

}

#' Get a chunk of a Feedly "stream"

#'

#' For the purposes of this short example, consider a

#' "stream" to be all the historical items in a feed.

#' (Note: the definition is more complex than that)

#'

#' Max "page size" (mad numbner of items returned in a single call)

#' is 1,000. For example simplicity, there's a blanket assumption

#' that if `continuation` is actually present, the caller is

#' savvy and asked for a large number of items (e.g. 10,000).

#' Therefore, assume we're paging by the thousands.

#'

#' @md

#' @param feed\_id the id of the stream (for this examplea feed id)

#' @param ct numnber of items to retrieve (API will only return 1,000

#' items for a single response and populate `continuation`

#' with a value that should be passed to subsequent calls

#' to page through the results; `ct` will be reset to 1,000

#' internally if this is the case)

#' @param continuation see `ct`

#' @references

#' @return for this example, an ugly `list`

feedly\_stream <- function(stream\_id, ct=100L, continuation=NULL) {

ct <- as.integer(ct)

if (!is.null(continuation)) ct <- 1000L

httr::GET(

url = "https://cloud.feedly.com/v3/streams/contents",

httr::add\_headers(

`Authorization` = sprintf("OAuth %s", .feedly\_token())

),

query = list(

streamId = stream\_id,

count = ct,

continuation = continuation

)

) -> res

httr::stop\_for\_status(res)

res <- httr::content(res, as="text")

res <- jsonlite::fromJSON(res)

res

}

We’ll grab 10,000 Feedly entries for the R-Bloggers feed stream:

r\_bloggers\_feed\_id <- "feed/http://feeds.feedburner.com/RBloggers"

rb\_stream <- feedly\_stream(r\_bloggers\_feed\_id, 10000L)

# preallocate space

streams <- vector("list", 10)

streams[1L] <- list(rb\_stream)

# gotta catch'em all!

idx <- 2L

while(length(rb\_stream$continuation) > 0) {

cat(".", sep="") # poor dude's progress par

feedly\_stream(

stream\_id = r\_bloggers\_feed\_id,

ct = 1000L,

continuation = rb\_stream$continuation

) -> rb\_stream

streams[idx] <- list(rb\_stream)

idx <- idx + 1L

}

cat("\n")

For those who aren’t used to piecing together bits from API’s like this (and for those who do not have a Pro account, those who didn’t want to write OAuth code or those who don’t use Feedly and cannot reproduce the post example), here’s some dissection:

str(streams, 1)

## List of 12

## $ :List of 7

## $ :List of 7

## $ :List of 7

## $ :List of 7

## $ :List of 7

## $ :List of 7

## $ :List of 7

## $ :List of 7

## $ :List of 7

## $ :List of 7

## $ :List of 7

## $ :List of 6 # No "continuation" in this one

str(streams[[1]], 1)

## List of 7

## $ id : chr "feed/http://feeds.feedburner.com/RBloggers"

## $ title : chr "R-bloggers"

## $ direction : chr "ltr"

## $ updated : num 1.52e+12

## $ alternate :'data.frame': 1 obs. of 2 variables:

## $ continuation: chr "15f457e2b66:160d6e:8cbd7d4f"

## $ items :'data.frame': 1000 obs. of 22 variables:

glimpse(streams[[1]]$items)

## Observations: 1,000

## Variables: 22

## $ id "XGq6cYRY3hH9/vdZr0WOJiPdAe0u6dQ2ddUFEsTqP10=\_1628f55fc26:7feb...

## $ keywords ["R bloggers", "R bloggers", "R bloggers", "R bloggers", "R b...

## $ originId "https://tjmahr.github.io/ridgelines-in-bayesplot-1-5-0-releas...

## $ fingerprint "f96c93f7", "9b2344db", "ca3762c8", "980635d0", "fbd60fac", "6...

## $ content c("

"Ridgelines in bayesplot 1.5.0", "Mathematical art in R", "R a...

## $ published 1.522732e+12, 1.522796e+12, 1.522714e+12, 1.522714e+12, 1.5227...

## $ crawled 1.522823e+12, 1.522809e+12, 1.522794e+12, 1.522793e+12, 1.5227...

## $ canonical [ c("feed/http://feeds.feedburner.com/RBloggers", "feed/h...

## $ author "Higher Order Functions", "David Smith", "R Views", "rOpenSci ...

## $ alternate [ c("At the end of March, Jonah Gabry and I released\nbay...

## $ visual c("feedly-nikon-v3.1", "feedly-nikon-v3.1", "feedly-nik...

## $ unread TRUE, TRUE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, F...

## $ categories [ 9, 37, 52, 15, 78, 35, 31, 9, 28, 2, 21, 8, 25, 11, 21, 29, 12...

## $ engagementRate 0.41, 1.37, 1.58, 0.45, 2.23, 0.97, 0.84, 0.23, 0.72, 0.05, 0....

## $ recrawled NA, NA, NA, NA, NA, NA, NA, NA, 1.522807e+12, NA, NA, NA, NA, ...

## $ tags [NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, ...

## $ decorations c("NA", "NA", "NA", "NA", "NA", "NA", "NA", "NA", "NA",...

## $ enclosure [NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, ...

That entries structure is defined in the Feedly API docs:

An entry is the atomic unit of content in the Feedly Cloud.

**Get the content of an entry**

GET /v3/entries/:entryId

(Authorization is **optional**)

**Response**

Status: 200 OK

{

"id"**:** "Xne8uW/IUiZhV1EuO2ZMzIrc2Ak6NlhGjboZ+Yk0rJ8=\_1523699cbb3:2aa0463:e47a7aef",

"keywords"**:** [

"TC",

"Microsoft",

"Internet-Explorer",

"Windows 10",

"Microsoft Edge",

"PCs"

],

"originId"**:** "http://techcrunch.com/?p=1261251",

"fingerprint"**:** "17c5dd0d",

"recrawled"**:** 1452618026719,

"thumbnail"**:** [

{

"url"**:** "https://tctechcrunch2011.files.wordpress.com/2016/01/microsoft-internet-explorer-10.png"

}

],

"title"**:** "Microsoft Today Ends Support For Windows 8, Old Versions Of Internet Explorer",

"published"**:** 1452614967000,

"crawled"**:** 1452614994867,

"summary"**:** {

"content"**:** "<img height=\"382\" alt=\"Microsoft-Internet-Explorer-10\" width=\"680\" class=\"wp-post-image\" src=\"https://tctechcrunch2011.files.wordpress.com/2016/01/microsoft-internet-explorer-10.png?w=680\"> Microsoft’s push towards Windows 10 continues. Today, Microsoft is ending support for Windows 8, as well as older versions of its Internet Explorer web browser, IE 8, IE 9, and IE 10. For end users, that doesn’t mean the software instantly becomes non-functional, but that it will longer be updated with bug fixes or other security patches.",

"direction"**:** "ltr"

},

"alternate"**:** [

{

"href"**:** "http://feedproxy.google.com/~r/Techcrunch/~3/iEm1aA\_M\_dw/",

"type"**:** "text/html"

}

],

"canonical"**:** [

{

"href"**:** "http://techcrunch.com/2016/01/12/microsoft-today-ends-support-for-windows-8-old-versions-of-internet-explorer/?ncid=rss",

"type"**:** "text/html"

}

],

"enclosure"**:** [

{

"href"**:** "https://tctechcrunch2011.files.wordpress.com/2016/01/microsoft-internet-explorer-10.png",

"type"**:** "image/png"

},

{

"href"**:** "http://tctechcrunch2011.files.wordpress.com/2016/01/microsoft-internet-explorer-10.png?w=150"

},

{

"href"**:** "http://2.gravatar.com/avatar/5225bb627e112543aa03bf3b2958be3f?s=96&d=identicon&r=G"

},

{

"href"**:** "https://tctechcrunch2011.files.wordpress.com/2016/01/edge\_phase2\_banner\_cortana\_1400px.jpg?w=680"

}

],

"author"**:** "Sarah Perez",

"origin"**:** {

"streamId"**:** "feed/http://feeds.feedburner.com/Techcrunch",

"title"**:** "TechCrunch",

"htmlUrl"**:** "http://techcrunch.com"

},

"visual"**:** {

"url"**:** "https://tctechcrunch2011.files.wordpress.com/%2F2016%2F01%2Fmicrosoft-internet-explorer-10.png",

"width"**:** 3000,

"height"**:** 1687,

"contentType"**:** "image/png"

},

"unread"**:** **false**,

"categories"**:** [

{

"id"**:** "user/c805fcbf-3acf-4302-a97e-d82f9d7c897f/category/Tech",

"label"**:** "Tech"

},

{

"id"**:** "user/c805fcbf-3acf-4302-a97e-d82f9d7c897f/category/global.must",

"label"**:** "Must Read"

}

],

"tags"**:** [

{

"id"**:** "user/c805fcbf-3acf-4302-a97e-d82f9d7c897f/tag/global.saved"

},

{

"id"**:** "user/c805fcbf-3acf-4302-a97e-d82f9d7c897f/tag/Microsoft",

"label"**:** "Microsoft"

}

],

"engagement"**:** 1476,

"engagementRate"**:** 7.88,

"updated"**:** 1452614967000,

"priorities"**:** [

{

"id"**:** "user/c805fcbf-3acf-4302-a97e-d82f9d7c897f/priority/22d2aead-dc47-40d4-bc4c-2301313aeb6c",

"label"**:** "Microsoft",

"searchTerms"**:** {

"parts"**:** [

{

"id"**:** "nlp/f/entity/wd:2283",

"label"**:** "Microsoft"

}

]

},

"actionTimestamp"**:** 1575630453903,

"streamId"**:** "user/c805fcbf-3acf-4302-a97e-d82f9d7c897f/category/Tech",

"streamLabel"**:** "Tech"

}

],

"commonTopics"**:** [

{

"type"**:** "topic",

"id"**:** "nlp/f/topic/487",

"label"**:** "cloud",

"score"**:** 0.981,

"salienceLevel"**:** "about"

}

],

"entities"**:** [

{

"disambiguated"**:** **true**,

"type"**:** "consumerGood",

"id"**:** "nlp/f/entity/wd:18698690",

"label"**:** "Microsoft Edge",

"mentions"**:** [

{

"text"**:** "microsoft edge"

},

{

"text"**:** "edge"

}

],

"salienceLevel"**:** "about"

},

{

"disambiguated"**:** **true**,

"type"**:** "org",

"id"**:** "nlp/f/entity/gz:org:microsoft",

"label"**:** "Microsoft",

"mentions"**:** [

{

"text"**:** "Microsoft"

}

],

"salienceLevel"**:** "about"

}

],

"memes"**:** [

{

"id"**:** "topic/tech/meme/c805fcbf-3acf-4302-a97e-d82f9d7c897f",

"label"**:** "Microsoft Explorer 10 Support End",

"score"**:** 1,

"featured"**:** **true**

}

],

"leoSummary"**:** {

"sentences"**:** [

{

"position"**:** 27,

"score"**:** 0.145,

"text"**:** "Today, Microsoft is ending support for Windows 8, as well as older versions of its Internet Explorer web browser, IE 8, IE 9, and IE 10."

}

]

}

}

Entries roughly follow the Atom format. Here is a list of fields you can expect to receive:

**id**

**string** the unique, immutable ID for this particular article.

**title**

*Optional* **string** the article’s title. This string does not contain any HTML markup.

**content**

*Optional* **content object** the article content. This object typically has two values: “content” for the content itself, and “direction” (“ltr” for left-to-right, “rtl” for right-to-left). The content itself contains sanitized HTML markup.

**summary**

*Optional* **content object** the article summary. See the content object above.

**author**

*Optional* **string** the author’s name

**crawled**

**timestamp** the immutable timestamp, in ms, when this article was processed by the feedly Cloud servers.

**recrawled**

*Optional* **timestamp** the timestamp, in ms, when this article was re-processed and updated by the feedly Cloud servers.

**published**

**timestamp** the timestamp, in ms, when this article was published, as reported by the RSS feed (often inaccurate).

**updated**

*Optional* **timestamp** the timestamp, in ms, when this article was updated, as reported by the RSS feed

**alternate**

*Optional* **link object array** a list of alternate links for this article. Each link object contains a media type and a URL. Typically, a single object is present, with a link to the original web page.

**origin**

*Optional* **origin object** the feed from which this article was crawled. If present, “streamId” will contain the feed id, “title” will contain the feed title, and “htmlUrl” will contain the feed’s website.

**keywords**

*Optional* **string array** a list of keyword strings extracted from the RSS entry.

**visual**

*Optional* **visual object** an image URL for this entry. If present, “url” will contain the image URL, “width” and “height” its dimension, and “contentType” its MIME type.

**unread**

**boolean** was this entry read by the user? If an Authorization header is not provided, this will always return false. If an Authorization header is provided, it will reflect if the user has read this entry or not.

**tags**

*Optional* **tag object array** a list of tag objects (“id” and “label”) that the user added to this entry. This value is only returned if an Authorization header is provided, and at least one tag has been added. If the entry has been explicitly marked as read (not the feed itself), the “global.read” tag will be present.

**categories**

**category object array** a list of category objects (“id” and “label”) that the user associated with the feed of this entry. This value is only returned if an Authorization header is provided.

**engagement**

*Optional* **integer** an indicator of how popular this entry is. The higher the number, the more readers have read, saved or shared this particular entry. This value changes over time.

**engagementRate**

*Optional* **float** a normalized indicator for the relative popularity of this entry compared to past data from the same source. A value below 1.0 indicates this entry is less popular, on average. A value above 1.0 indicates this entry is more popular. Because this value is normalized, it can be used to compare entries from other sources, and is used for engagement ranking. This value changes over time.

**actionTimestamp**

*Optional* **timestamp** for tagged articles, contains the timestamp when the article was tagged by the user. This will only be returned when the entry is returned through the streams API.

**enclosure**

*Optional* **link object array** a list of media links (videos, images, sound etc) provided by the feed. Some entries do not have a summary or content, only a collection of media links.

**fingerprint**

**string** the article fingerprint. This value might change if the article is updated.

**originId**

**string** the unique id of this post in the RSS feed (not necessarily a URL!)

**sid**

*Optional* **string** an internal search id.

**priorities**

*Optional* **priority object array** a list of priority filters that match this entry (pro+ and team only).

**memes**

*Optional* **meme object array** a list of [memes](https://developer.feedly.com/v3/memes): clusters of entries from popular sources that are about the same subject. The meme id can be used to retrieve the other articles about the same subject.

**leoSummary**

*Optional* **summary object** For pro+ and enterprise sources, Feedly will extract one or two important sentences from the entry content, to be used for summary or highlights. The text does not include any HTML tags.

**commonTopics**

*Optional* **topic object array** a list of detected topics in this article. This feature is only available for pro+ and enterprise feeds. Salience level can either be about (if the article is about this topic), or mention (if the article only mentions this topic)

**entities**

*Optional* **entity object array** a list of detected entities in this article. This feature is only available for pro+ and enterprise feeds. mentions will list the text fragments that refer to each entity.

**related**

*Optional* **related entries object array** a list of related or similar entries. This data is only available for pro+ and enterprise users, if similar=true is passed to the streams API.

**Get the content for a dynamic list of entries**

POST /v3/entries/.mget

(Authorization is **optional**)

**Input**

[

"gRtwnDeqCDpZ42bXE9Sp7dNhm4R6NsipqFVbXn2XpDA=\_13fb9d6f274:2ac9c5:f5718180",

"9bVktswTBLT3zSr0Oy09Gz8mJYLymYp71eEVeQryp2U=\_13fb9d1263d:2a8ef5:db3da1a7"

]

The number of entry ids you can pass as an input is limited to 1,000.

**Response**

Status: 200 OK

[

{

"id"**:** "entryId",

"unread"**:** **true**,

"categories"**:** [

{

"id"**:** "user/c805fcbf-3acf-4302-a97e-d82f9d7c897f/category/tech",

"label"**:** "tech"

}

],

"tags"**:** [

{

"id"**:** "user/c805fcbf-3acf-4302-a97e-d82f9d7c897f/tag/inspiration",

"label"**:** "inspiration"

}

],

"title"**:** "NBC's reviled sci-fi drama 'Heroes' may get a second lease on life as Xbox Live exclusive",

"keywords"**:** [

"NBC",

"Sci-fi"

],

"published"**:** 1367539068016,

"updated"**:** 1367539068016,

"crawled"**:** 1367539068016,

"alternate"**:** [

{

"href"**:** "http://www.theverge.com/2013/4/17/4236096/nbc-heroes-may-get-a-second-lease-on-life-on-xbox-live",

"type"**:** "text/html"

}

],

"content"**:** {

"direction"**:** "ltr",

"content"**:** "..."

},

"author"**:** "Nathan Ingraham",

"origin"**:** {

"streamId"**:** "feed/http://www.theverge.com/rss/full.xml",

"title"**:** "The Verge - All Posts",

"htmlUrl"**:** "http://www.theverge.com/"

},

"engagement"**:** 12

},

{

"id"**:** "entryId2",

"unread"**:** **true**,

"categories"**:** [

{

"id"**:** "user/c805fcbf-3acf-4302-a97e-d82f9d7c897f/category/tech",

"label"**:** "tech"

}

],

"tags"**:** [

{

"id"**:** "user/c805fcbf-3acf-4302-a97e-d82f9d7c897f/tag/inspiration",

"label"**:** "inspiration"

}

],

"title"**:** "Senate rejects bipartisan gun control measure for background checks despite broad public support",

"keywords"**:** [

"gun control"

],

"published"**:** 1367539068016,

"updated"**:** 1367539068016,

"crawled"**:** 1367539068016,

"alternate"**:** [

{

"href"**:** "http://www.theverge.com/2013/4/17/4236136/senate-rejects-gun-control-amendment",

"type"**:** "text/html"

}

],

"content"**:** {

"direction"**:** "ltr",

"content"**:** "...html content..."

},

"author"**:** "T.C. Sottek",

"origin"**:** {

"streamId"**:** "feed/http://www.theverge.com/rss/full.xml",

"title"**:** "The Verge - All Posts",

"htmlUrl"**:** "http://www.theverge.com/"

},

"engagement"**:** 24,

"engagementRate"**:** 1.23

}

]

**Create and tag an entry**

This call is useful to inject entries not coming from a feed, into a user’s account. The entries created will only be available through the tag streams of the respective tags passed.

POST /v3/entries/

(Authorization is **required**)

**Input**

{

"tags"**:** [

{

"id"**:** "user/c805fcbf-3acf-4302-a97e-d82f9d7c897f/tag/global.saved"

},

{

"id"**:** "user/c805fcbf-3acf-4302-a97e-d82f9d7c897f/tag/inspiration",

"label"**:** "inspiration"

}

],

"title"**:** "NBC's reviled sci-fi drama 'Heroes' may get a second lease",

"keywords"**:** [

"NBC",

"sci-fi"

],

"alternate"**:** [

{

"href"**:** "http://www.theverge.com/2013/4/17/4236096/nbc-heroes-may-get-a-second-lease-on-life-on-xbox-live",

"type"**:** "text/html"

}

],

"content"**:** {

"direction"**:** "ltr",

"content"**:** "...html content the user wants to associate with this entry.."

},

"author"**:** "Nathan Ingraham",

"origin"**:** {

"title"**:** "The Verge - All Posts",

"htmlUrl"**:** "http://www.theverge.com/"

}

}

The following fields must be present:

* **title**
* **content** or **summary** or **enclosure**
* **origin**.
* **alternate**
* **published**

If values are not provided, **crawled**, **published** and **updated** timestamps will be automatically set. The following fields will be ignored: **engagement**, **unread**, **visual**, **actionTimestamp**, **categories**. The entry id will be automatically generated.

**Response**

Status: 200 OK

[

"gRtwnDeqCDpZ42bXE9Sp7dNhm4R6NsipqFVbXn2XpDA=\_13fb9d6f274:2ac9c5:f5718180",

"9bVktswTBLT3zSr0Oy09Gz8mJYLymYp71eEVeQryp2U=\_13fb9d1263d:2a8ef5:db3da1a7"

]

We’ll extract the bits we want to use for the rest of the post and clean it up a bit:

map\_df(streams, ~{

select(.x$items, title, author, published, engagement) %>%

mutate(published = anytime::anydate(published / 1000)) %>% # overly-high-resolution timestamp

tbl\_df()

}) -> xdf

glimpse(xdf)

## Observations: 11,421

## Variables: 4

## $ title "Ridgelines in bayesplot 1.5.0", "Mathematical art in R", "R and T...

## $ author "Higher Order Functions", "David Smith", "R Views", "rOpenSci - op...

## $ published 2018-04-03, 2018-04-03, 2018-04-02, 2018-04-02, 2018-04-03, 2018-...

## $ engagement 9, 37, 52, 15, 78, 35, 31, 9, 28, 2, 21, 8, 25, 11, 21, 29, 12, 11...

Using an arbitrary “10,000” extract didn’t give us full months:

range(xdf$published)

## [1] "2013-05-31" "2018-04-03"

so we’ll filter out the incomplete bits and add in some additional temporal metadata:

xdf %>%

filter(

published > as.Date("2013-05-31"), # complete months

published < as.Date("2018-04-01")

) %>%

mutate(

year = as.integer(lubridate::year(published)),

month = lubridate::month(published, label=TRUE, abbr=TRUE),

wday = lubridate::wday(published, label=TRUE, abbr=TRUE),

ym = as.Date(format(published, "%Y-%m-01"))

) -> xdf

I’m only going to do some light analysis work with engagement data (how “popular” a post was) but the *full post summary and body content* is available in the data dump you’re going to get at the end. That means enterprising folk can do some tidy text mining to cluster away some additional insights.

Thankfully, there’s not a ton of missing engagement data:

sum(is.na(xdf$engagement)) / nrow(xdf)

## [1] 0.06506849

broom::tidy((summary(xdf$engagement)))

## minimum q1 median mean q3 maximum na

## 1 0 5 20 69.27219 75 4785 741

Let’s look at post count over time, first:

count(xdf, ym) %>%

arrange(ym) %>%

ggplot(aes(ym, n)) +

ggforce::geom\_bspline0(color="lightslategray") +

scale\_x\_date(expand=c(0,0.5)) +

labs(

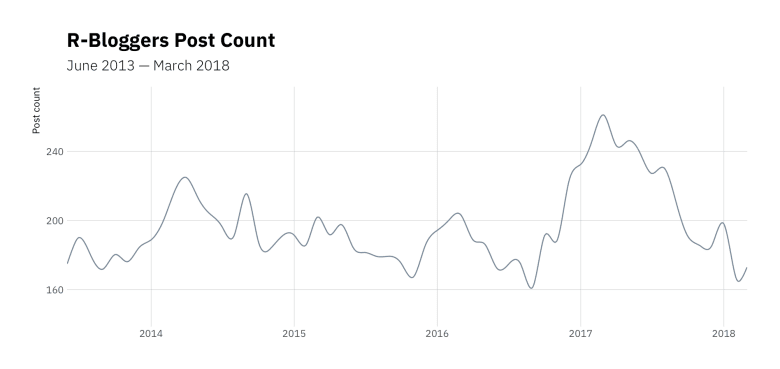
x=NULL, y="Post count",

title="R-Bloggers Post Count",

subtitle="June 2013 — March 2018"

) +

theme\_ipsum\_ps(grid="XY")

[](https://i0.wp.com/rud.is/b/wp-content/uploads/2018/04/rb-post-count.png?ssl=1)

It’ll be interesting to watch that over this year and compare 2017 to 2018 given how “hot” 2017 seems to have been. To turn a Mythbuster phrase: a neat “try this at home” exercise would be to tease out some “whys” for various spikes (which likely means some post content spelunking).

Let’s see if any days are more popular than others:

count(xdf, wday) %>%

ggplot(aes(wday, n)) +

geom\_col(fill="lightslategray", width=0.65) +

scale\_y\_comma() +

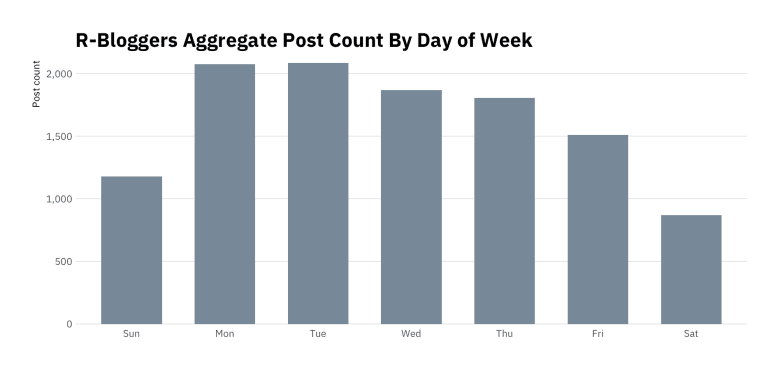
labs(

x=NULL, y="Post count",

title="R-Bloggers Aggregate Post Count By Day of Week"

) +

theme\_ipsum\_ps(grid="Y")

[](https://i1.wp.com/rud.is/b/wp-content/uploads/2018/04/rb-post-count-dow.png?ssl=1)

Weekends are sleepy and there are some “go-getters” at the beginning of the week. More “try this at home” would be to see if any individuals have “patterns” by day of week (or even time of day, since that’s also available in the published time stamp).

The summary() above told us we have a pretty skewed engagement distribution, but it’s always nice to visualise just how bad it is:

ggplot(xdf, aes(engagement)) +

geom\_density(aes(y=calc(count)), fill="lightslategray", alpha=2/3) +

scale\_x\_comma() +

scale\_y\_comma() +

labs(

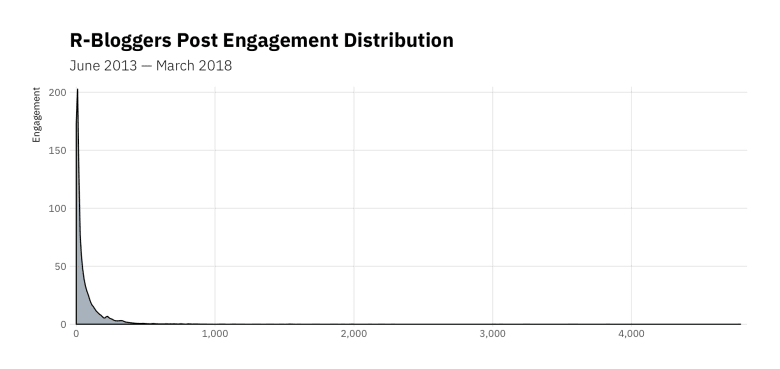
x=NULL, y="Engagement",

title = "R-Bloggers Post Engagement Distribution",

subtitle = "June 2013 — March 2018"

) +

theme\_ipsum\_ps(grid="XY")

[](https://i0.wp.com/rud.is/b/wp-content/uploads/2018/04/rb-engagement.png?ssl=1)

That graph is the story of my daily life dealing with internet data. Couldn’t even get a break when trying to have some fun. #sigh

We’ll close with the “all time top 10” based on total engagement:

count(xdf, author, wt=engagement, sort=TRUE)

## # A tibble: 1,065 x 2

## author n

## 1 David Smith 87381

## 2 Tal Galili 29302

## 3 Joseph Rickert 16846

## 4 DataCamp Blog 14402

## 5 DataCamp 14208

## 6 John Mount 13274

## 7 Francis Smart 8506

## 8 hadleywickham 8129

## 9 hrbrmstr 7855

## 10 Sharp Sight Labs 7620

## # ... with 1,055 more rows