Group Homework #8

Adrian Perez, Hoik Jang, James Chen, Steven Oliver

2018 10 24

Group Project #5

Introduction

In the last week, we decided to change our dataset from the cosmic particle data to data on user comments on the social media site Reddit. We did this because we thought that our old dataset was too technical to be used creatively for a data science project. In particular, we didn't have too many fields to work with, and those we had were limited in complexity.

For the unfamiliar, Reddit operates as a collection of communities known as subreddits. These are message boards oriented around a topic of interest (e.g., r/NBA, r/politics, r/science). On a message board, users interact by either making posts of content (images, gifs, or a text discussion) or commenting on other users' posts. In addition, users can also vote up or down other users' comments and posts, depending on whether they believe the material to be interesting or relevant (or agreeable).

We would like to analyze the patterns of user comment activity in a multi-month span on Reddit. The ideal would be investigating how closely linked the largest subreddits on the site are to each other by tracking the spread of internet memes across those communities. However, due to the richness of the dataset, we can also look at many other issues. For instance, we can see whether the level of community moderator activity affects the distribution of user scores of comments, or the fluctuations in user activity across subreddits by time of day.

Our new primary dataset of user comments contains the following fields:

```
## opening file input connection.
```

```
##
 Found 500 records...
 Found 1000 records...
 Found 1500 records...
 Found 2000 records...
 Found 2500 records...
 Found 3000 records...
 Found 3500 records...
 Found 4000 records...
 Found 4500 records...
 Found 5000 records...
 Found 5500 records...
 Found 6000 records...
 Found 6500 records...
 Found 7000 records...
 Found 7500 records...
 Found 8000 records...
 Found 8500 records...
 Found 9000 records...
 Found 9500 records...
 Found 10000 records...
 Imported 10000 records. Simplifying...
```

```
## closing file input connection.
```

```
##  [1] "author"                "author_flair_css_class"
##  [3] "author_flair_text"     "body"
##  [5] "can_gild"              "controversiality"
##  [7] "created_utc"           "distinguished"
##  [9] "edited"                "gilded"
## [11] "id"                    "is_submitter"
## [13] "link_id"               "parent_id"
## [15] "permalink"             "retrieved_on"
## [17] "score"                 "stickied"
## [19] "subreddit"             "subreddit_id"
## [21] "author_cakeday"
```

The most relevant fields for us are author (the user submitting the comment), body (the comment text), distinguished (whether or not user is a moderator of the subreddit the comment is posted in), score, and subreddit.

You might notice that this is a tiny dataset of 10000 observations. This is because we currently have a bottleneck in memory and hard disk space, as one month of Reddit comments is about 45 GB in size. We are pulling our data from a publicly posted web scrape, for which the smallest files are daily scrapes of ~ 2 GB sized JSON files. We tried loading these into R and exporting the dataframe to our database, but they expand greatly when loaded into R and are too big for 16 GB of RAM.

We expect to set up a system for splitting our data into manageable chunks via the jsonlite library in R in the next few days. Once we do that, we can import them into our database piecemeal, while also filtering them at the same time. For now though, we'll be working with a json of 10000 sample records provided by the original web scraper.

```
dbRemoveTable(conn = dcon, name = "Sample_Data")
dbWriteTable(conn = dcon, name = "Sample_Data", sample_data,
             append=FALSE, row.names = FALSE)
```

As a temporary secondary data source, we used a table called subreddits_basic, which contained the IDs and subscriber counts for each of subreddits on Reddit. As this was a CSV, we loaded it into a new database using DB Browser for SQLITE.

Querying Data into Subdata

For our full-sized data sets, we will need to filter the records drastically so they can fit in memory on R. The most obvious way to do this would be to limit our sample to those comments from the X most popular subreddits. So we created three reference tables of the 100, 500, and 1000 most popular subreddits by number of subscribers. This way we can filter out the relevant comments with a simple inner join.

Subdata #1

```
dbListFields(dcon, "Sample_Data")
```

```
##  [1] "author"                "author_flair_css_class"
##  [3] "author_flair_text"     "body"
##  [5] "can_gild"              "controversiality"
##  [7] "created_utc"           "distinguished"
##  [9] "edited"                "gilded"
## [11] "id"                    "is_submitter"
## [13] "link_id"               "parent_id"
```

```
## [15] "permalink"              "retrieved_on"
## [17] "score"                  "stickied"
## [19] "subreddit"              "subreddit_id"
## [21] "author_cakeday"
```

```
dbListFields(dcon, "subreddits_basic")
```

```
## [1] "base10id"            "redditbase36id"      "creationepoch"
## [4] "subredditname"       "numberofsubscribers"
```

```
query <- "
SELECT subredditname, numberofsubscribers
FROM subreddits_basic
WHERE numberofsubscribers !='None'
ORDER BY numberofsubscribers DESC
LIMIT 100;"
res <- dbSendQuery(dcon, query)
subreddits100 <-dbFetch(res, -1)
dbClearResult(res)
dbRemoveTable(conn = dcon, name = "subreddits100")
dbWriteTable(conn = dcon, name = "subreddits100", subreddits100,
             append = FALSE, row.names = FALSE)
head(subreddits100)
```

```
##   subredditname numberofsubscribers
## 1         funny            19467413
## 2     AskReddit            19177858
## 3 todayilearned            18755662
## 4     worldnews            18671986
## 5       science            18641755
## 6          pics            18592612
```

Subdata #2

```
query <- "
SELECT subredditname, numberofsubscribers
FROM subreddits_basic
WHERE numberofsubscribers !='None'
ORDER BY numberofsubscribers DESC
LIMIT 500;"
res <- dbSendQuery(dcon, query)
subreddits500 <-dbFetch(res, -1)
dbClearResult(res)
dbRemoveTable(conn = dcon, name = "subreddits500")
dbWriteTable(conn = dcon, name = "subreddits500", subreddits500,
             append = FALSE, row.names = FALSE)
head(subreddits500)
```

```
##   subredditname numberofsubscribers
## 1         funny            19467413
## 2     AskReddit            19177858
## 3 todayilearned            18755662
## 4     worldnews            18671986
## 5       science            18641755
## 6          pics            18592612
```

Subdata #3

```
query <- "
SELECT subredditname, numberofsubscribers
FROM subreddits_basic
WHERE numberofsubscribers !='None'
ORDER BY numberofsubscribers DESC
LIMIT 1000;"
res <- dbSendQuery(dcon, query)
subreddits1000 <-dbFetch(res, -1)
dbClearResult(res)
dbRemoveTable(conn = dcon, name = "subreddits1000")
dbWriteTable(conn = dcon, name = "subreddits1000", subreddits1000,
             append = FALSE, row.names = FALSE)

head(subreddits1000)
```

```
##   subredditname numberofsubscribers
## 1         funny            19467413
## 2     AskReddit            19177858
## 3 todayilearned            18755662
## 4     worldnews            18671986
## 5       science            18641755
## 6          pics            18592612
```

Working with Data

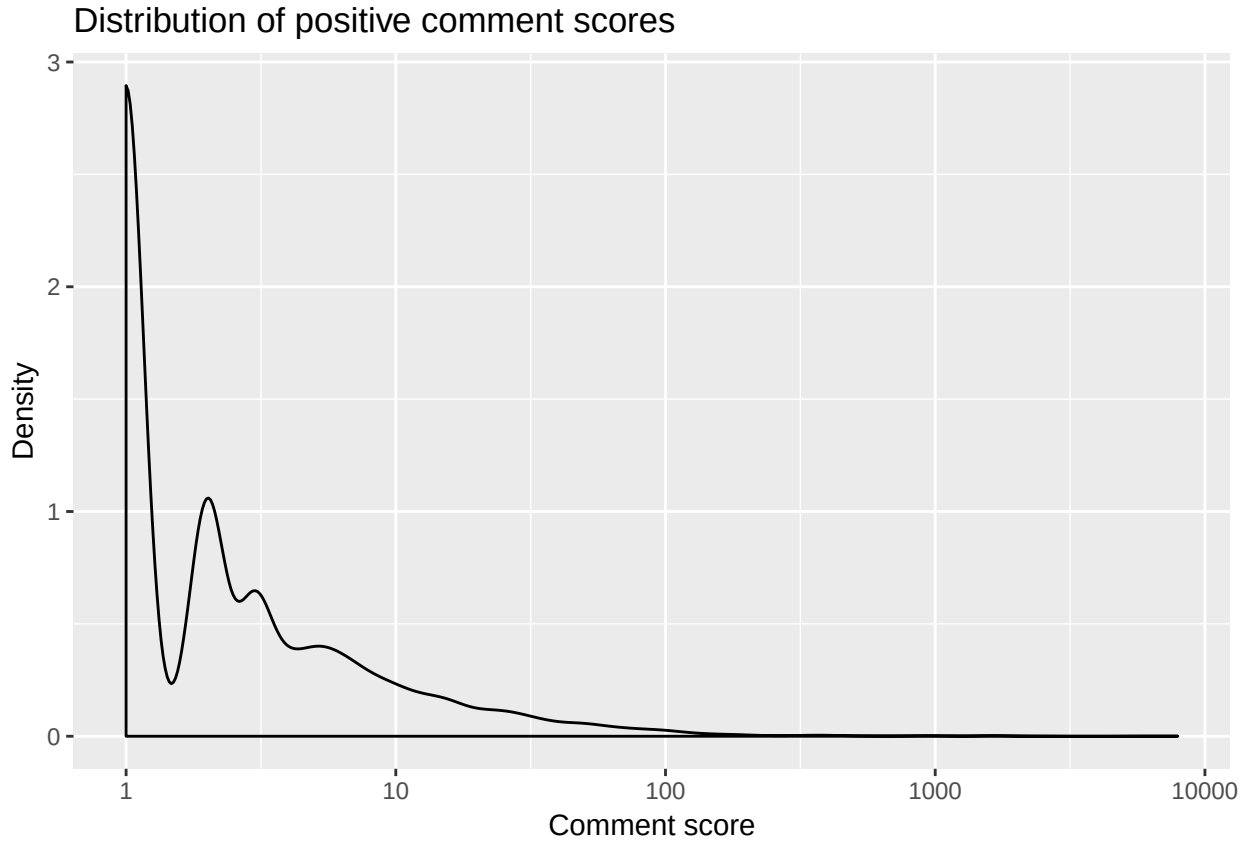
Here are a couple of sample visualizations of our data:

Distribution of user scores

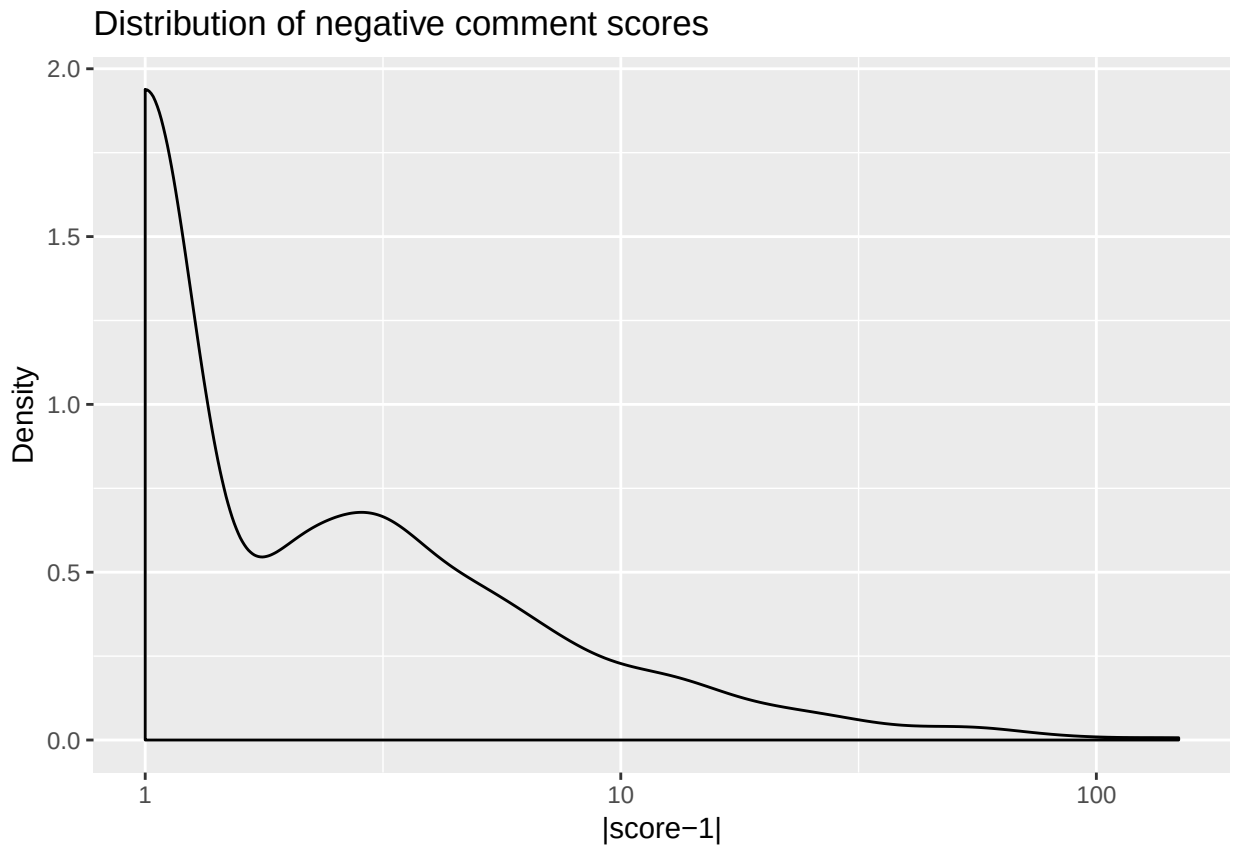
Since the scores are heavily skewed to the right, we need to use a logarithmic scale to show their distribution. But we need to keep in mind that user scores can be zero or negative. So we need to make a separate plot for those comments.

One feature of the data we can see immediately is most comments have scores of either 1 or 2. This is unsurprising, because the default score of any comment is 1, so these two scores capture the bulk of comments where the reaction is either neutral or only slightly positive for most users.

```
##
## Attaching package: 'dplyr'
## The following objects are masked from 'package:stats':
##
##     filter, lag
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```



Here is the same plot for comments with non-positive scores. Similarly to the discussion with the positive comments, the most common negative score seems to be 0, which is indicative of a slightly negative reaction from other users, since the default is 1.



Distribution of Subreddit Sizes by Subscriber Count

All subreddits

Loading the set of all 1 million subreddits into R. If this was too big to load into memory, we would've moved on to the next step of top 1000 subreddits directly.

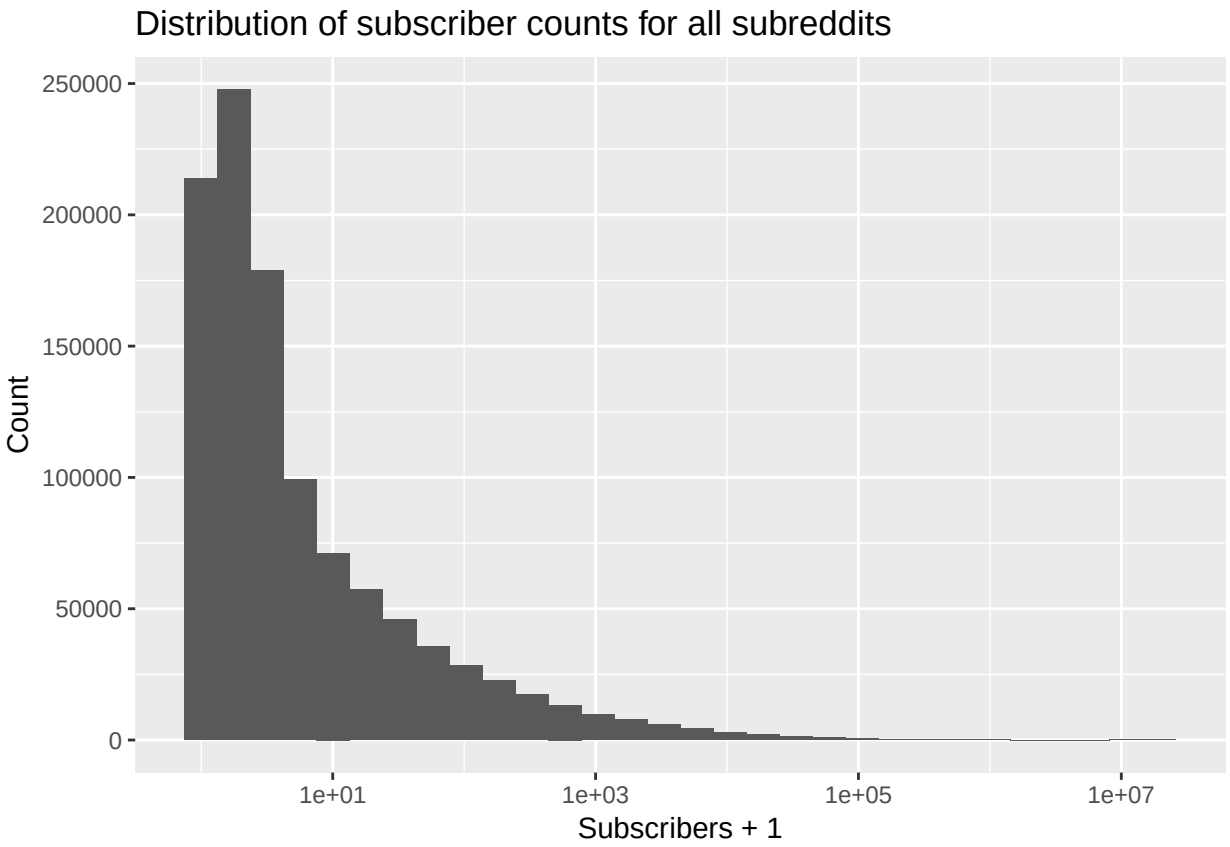
```
## # A tibble: 1 x 1
##         n
##     <int>
## 1 1067472
```

We can see that the distribution of subscribers is heavily skewed to the right, even with logarithmic scaling.

```
## Warning: Transformation introduced infinite values in continuous x-axis
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 12 rows containing non-finite values (stat_bin).
```

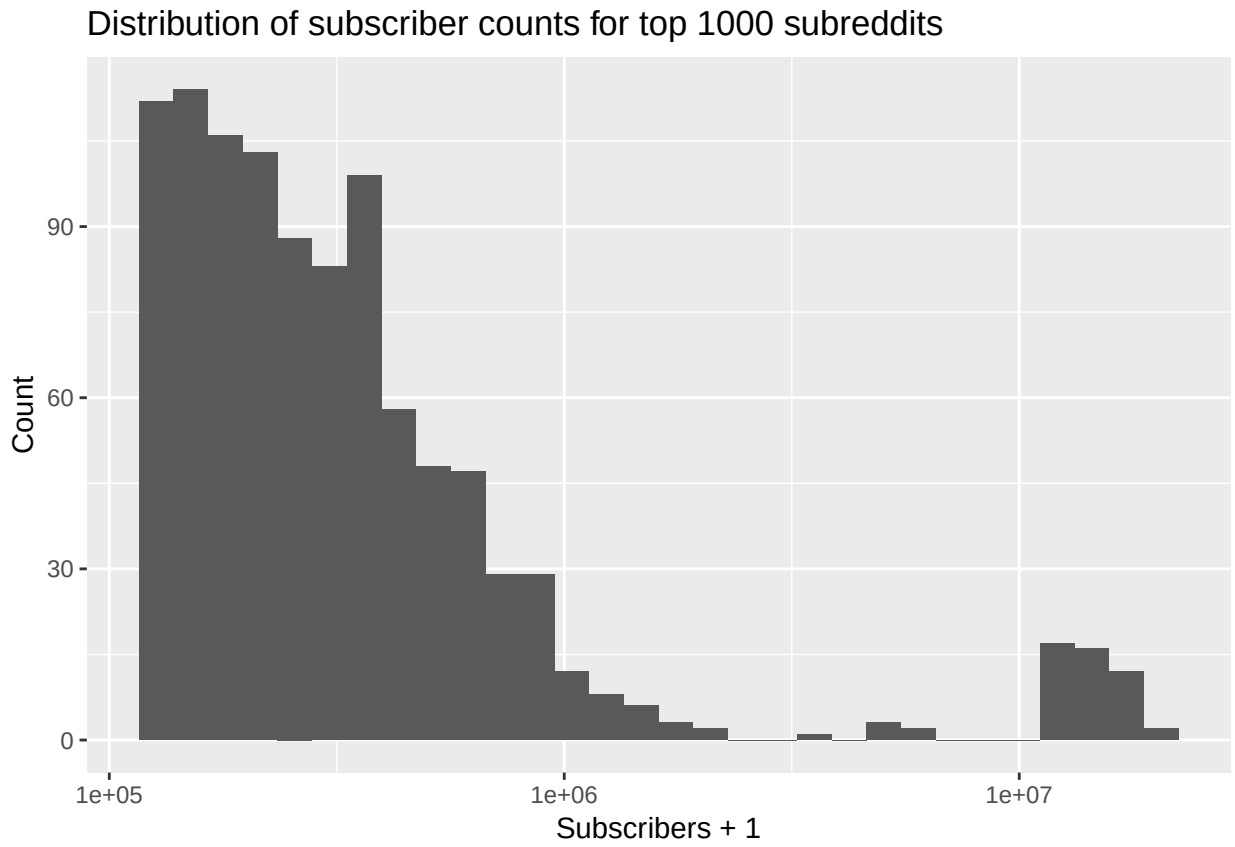


Top 1000 subreddits

When we focus on the top 1000 subreddits, we can see that there is actually a noticeable clump of outliers on the far right of the distribution. As it turns out, this is because there is a list of broad-interest subreddits (for example, r/news, r/sports, and r/art) to which Reddit automatically subscribes new users to acclimate them to the site.

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##   120377   170467   262010  1045351   451114 19467413
```

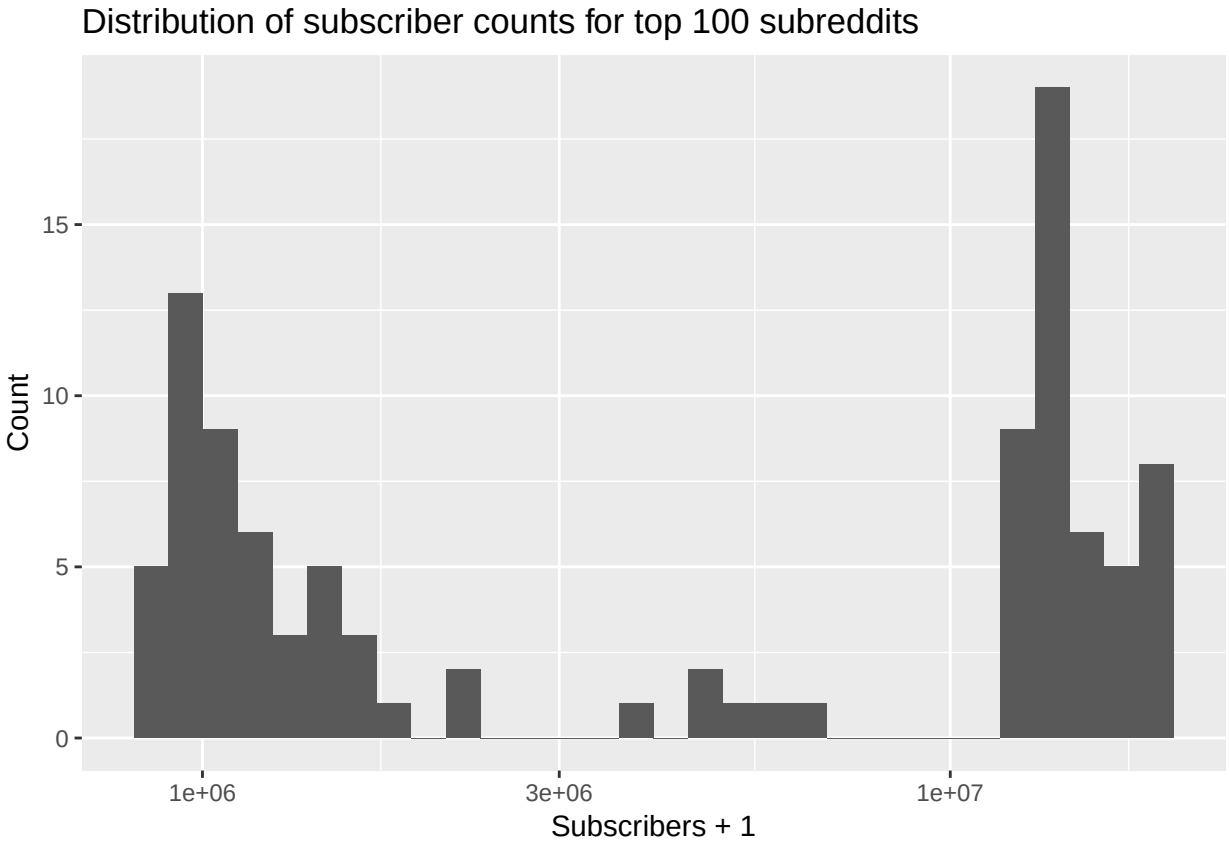
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



If we look at only the top 100 subreddits, we can see the same clustering phenomenon. Since Reddit occasionally rotates out less-popular default subreddits, it is likely that variation in the subscriber counts for the top subreddits is largely driven by the amount of time that they have been default subscriptions.

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##   882016  1091266  4996884  7788097 13475356 19467413
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```


Distribution of subscriber counts for top 100 subreddits
Count
15
10
5
0
1e+06
3e+06
1e+07
Subscribers + 1