````
---
title: "RedditExtractoR Example"
author: "Gagan S"
format: html
editor: visual
---

### Install RedditExtractoR

```{r}
install.packages("RedditExtractoR")

library(RedditExtractoR)
```

### Special Considerations

This package will limit the number of posts you can gather at any one period of time to 1000. It is also very slow, so I recommend setting your search criteria to a reasonable time frame (see below).

### `find_thread_urls`

This is the main function I use. It will return posts based on specified criteria:

-   keywords - this will search for posts containing that keyword across all of Reddit

-   subreddit - this will search for all of the posts within a specific subreddit

-   keyword + subreddit - if you specify both, it will search for all of posts with that keyword within a specific subreddit.

You can also specify how to sort posts as well as the time frame to search:

-   sort_by - allows you to sort posts you want by different criteria: if you are searching with key words, then you can sort by relevance, comments, new, hot, top; if you are not searching by keywords, then it must be one of: hot, new, top, rising.

-   period - the time period to search by it will return posts that fit your critieria made in the last: hour, week, month, year, and all

```{r}
example1 <- find_thread_urls(
  keywords = "crane",
  sort_by = "relevance",
  subreddit = "origami",
  period = "month"
)

head(example1)
```

The information that you'll get from this posts includes:

-   date_utc - date of the post in the utc format (year-month-day)

-   timestamp - a timestamp for the post

-   title - the title of the post

-   text - the body of the post

-   subreddit - what subreddit the post came from

-   comments - the number of comments for that post

-   url - the url for that post (this is important for the next function!)

### `get_thread_content`
````

Once you have gotten relevant posts, you can also get more information on that post, including the text for each comment on that post using this function. By default it will return a list with 2 data frames: 1. meta data for each post; 2. all of the comments in that post.

To set this up, you need to either create a separate data frame with the url column from above or specify that you using the url column using the \$ operator.

This takes even longer than the last function to run, so only use it if you really need comments!

```{r}
urls <- example1$url

example2 <- get_thread_content(urls)

example2 <- get_thread_content(example1$url)
```

```{r}
head(example2$threads)
```

In addition to all of the information you got from `find_threads_url` The "threads" data frame will return:

- score - how many points the post got
- upvotes
- downvotes
- up_ratio - up to down vote ratio
- total_awards - how many awards like gold, that post got
- golds - the number of reddit gold that post specifically got
- cross_posts - how many times that post got shared to other subreddits
- comments - the number of comments

```{r}
head(example2$comments)
```

the "comments" data frame will give you the following information:

- url - a url linking that comment to each post
- author - the username of who wrote each comment
- date - the date of the comment in UTC format
- timestamp
- score - points for each comment
- upvotes - upvotes for each comment
- downvotes - downvotes for each comment
- golds - the number of reddit gold that comment received
- comment - the actual comment text
- comment_id - links the comment to each post, it will start with 1 indicating the first comment within a specific post, and iterate to 1_1, 1_2, etc. indicating that it is the second, third, etc. comment for that specific post.