Steps to install R

1. R installation takes up to 150MB of disk space.
2. You can [download the R Windows installer](http://cran.r-project.org/bin/windows/base/) from CRAN (<https://cran.r-project.org/>); then double-click on the installer to install R as you would any Windows software.
3. If you have an account with Administrator privileges you will be able to install R in the Program Files area and to set all the optional registry entries; otherwise you will only be able to install R in your own file area. You may need to confirm that you want to proceed with installing a program from an ‘unknown’ or ‘unidentified’ publisher.
4. After installation you should choose a working directory for R. You will have a shortcut to Rgui.exe on your desktop and/or somewhere on the Start menu file tree. Right-click each shortcut, select Properties and change the ‘Start in’ field to your working directory.
5. You may also want to add command-line arguments at the end of the Target, for example --sdi --max-mem-size=1G. You can also set environment variables at the end of the Target field, for example R\_LIBS=p:/myRlib, and if you want to ensure that menus and messages are in English, LANGUAGE=en.

Steps to install RStudio

1. First, go to <http://www.rstudio.com/> and click Download now.
2. Next, click Download RStudio Desktop.
3. Next, click the link to the version of RStudio appropriate for your system. The first link is to the version that RStudio recommends for you. Clicking this link downloads RStudio to your computer.
4. Next, find the file that was downloaded and double click it. This will start the install process.
5. Click next to continue when the install wizard opens.
6. Click next to accept the default install location.
7. Click Install to accept the default start menu folder and install RStudio. Click Finish to close the wizard.

Steps to install a package in R

1. Choose Install Packages from the Packages menu.
2. Select a CRAN Mirror. (e.g. Norway)
3. Select a package. (e.g. boot)
4. Then use the library(*package*) function to load it for use. (e.g. library(boot))

Twitter Rate Limiting

* Per User or Per Application

Rate limiting of the API is primarily considered on a per-user basis — or more accurately described, per access token in your control. When using [application-only authentication](https://dev.twitter.com/oauth/application-only), rate limits are determined globally for the entire application. This limit is considered completely separately from per-user limits.

* 15 Minute Windows

Rate limits are divided into 15 minute intervals. There are two initial buckets available for GET requests: 15 calls every 15 minutes, and 180 calls every 15 minutes.

* Search

Search is rate limited at 180 queries per 15 minute window.

* HTTP Headers and Response Codes

HTTP headers provide pertinent data on where your application is at for a given rate limit on the method that you just utilized.

* X-Rate-Limit-Limit: the rate limit ceiling for that given request
* X-Rate-Limit-Remaining: the number of requests left for the 15 minute window
* X-Rate-Limit-Reset: the remaining window before the rate limit resets in UTC[epoch seconds](http://en.wikipedia.org/wiki/Unix_time)

When an application exceeds the rate limit for a given API endpoint, the Twitter API will now return an HTTP 429 [“Too Many Requests”](http://tools.ietf.org/html/rfc6585) response code.

To better predict the rate limits available to you, consider periodically using [GET application / rate\_limit\_status](https://dev.twitter.com/rest/reference/get/application/rate_limit_status). The rate limiting HTTP headers and resource’s response will indicate the rate limit status for the calling context — when using app-only auth, the limits will pertain to that auth context. When using user-based auth, the limits will pertain to the application-user context.

* GET and POST Request Limits

Rate limits on “reads” from the system are defined on a per user and per application basis, while rate limits on writes into the system are defined solely at the user level.

Lastly, there may be times in which the rate limit values that we return are inconsistent, or cases where no headers are returned at all. Perhaps memcache has been reset, or one memcache was busy so the system spoke to a different instance: the values may be inconsistent now and again. The system makes a best effort to maintain consistency, but errs towards giving an application extra calls if there is an inconsistency.

Tips to avoid being Rate Limited

The tips below are there to help you code defensively and reduce the possibility of being rate limited.

* Caching

Store API responses in your application or on your site if you expect a lot of use. For example, don’t try to call the Twitter API on every page load of your website landing page. Instead, call the API infrequently and load the response into a local cache. When users hit your website load the cached version of the results.

* Prioritize active users

If your site keeps track of many Twitter, consider only requesting data for users who have recently signed into your site.

* Adapt to the search results

If your application monitors a high volume of search terms, query less often for searches that have no results than for those that do. By using a back-off you can keep up to date on queries that are popular but not waste cycles requesting queries that very rarely change.

* Use application-only auth as a “reserve”

Requests using [Application-only authentication](https://dev.twitter.com/oauth/application-only) are evaluated in a separate context to an application’s per-user rate limits. For many scenarios, you may want to use this additional rate limit pool as a “reserve” for your typical user-based operations.

* Blacklisting

We ask that developers honor the rate limits. If you or your application abuse the rate limits, the account or app will be blacklisted. If an app is blacklisted it will be unable to get a response from the Twitter API. If you or your application has been blacklisted and you think there has been a mistake, you can use our [Platform Support forms](https://support.twitter.com/forms/platform).

* Streaming API

The Streaming API has rate limiting and access levels that are appropriate for long-lived connections. Leveraging the Streaming API is a great way to free-up your rate limits for more inventive uses of the Twitter API.

* Limits Per Window Per Resource

The API rate limit window duration is 15 minutes long. There is an [API Rate Limit: Chart](https://dev.twitter.com/rest/public/rate-limits) to see the limits by resource.

Creating a Twitter Application

To use Twitter Application Program Interfaces(APIs), you need OAuth access keys. To get Twitter Access keys, you need to create Twitter Application which is mandatory to access Twitter.

1. Go to <https://dev.twitter.com/apps/new> and log in, if necessary
2. Enter your Application Name, Description and your website address. You can leave the callback URL empty.
3. Accept the TOS, and solve the CAPTCHA.
4. Submit the form by clicking the Create your Twitter Application
5. Copy the consumer key (API key) and consumer secret from the screen into your application

After creating your Twitter Application, you have to give the access to your Twitter Account to use this Application. To do this, click the Create my Access Token.

In order to access the Twitter, that is to get recent tweets and twitter followers count, you need the four keys such as Consumer Key, Consumer Secret, Acess token, Access Token Secret.

To get all these keys, click the OAuth Tool tab in your Twitter Application and copy those keys and paste in your application.