

03-01: Packages

1 - Purpose

- Introduce R packages
- Connect to the NOAA/NCDC database using a package
- Get weather data from the NOAA/NCDC database

2 - Concepts

3 - The NOAA/NCDC Climate database

For the majority of the unit, we will be using weather data from the NOAA/NCDC governmental databases. The weather databases are free to use but you first need to get a *token*, which is similar to a password. So, before we write script to connect to the database, let's get the token.

You can [register for a token on this page](#). Just put in your email address and NOAA/NCDC will send a very random looking 32 character string to your email. This 32 character string is your token and allows you to access the climate database. Don't forget to check your SPAM folder for the email if you don't receive it right away. I have tested it a few times and got the email within 1 minute each time.

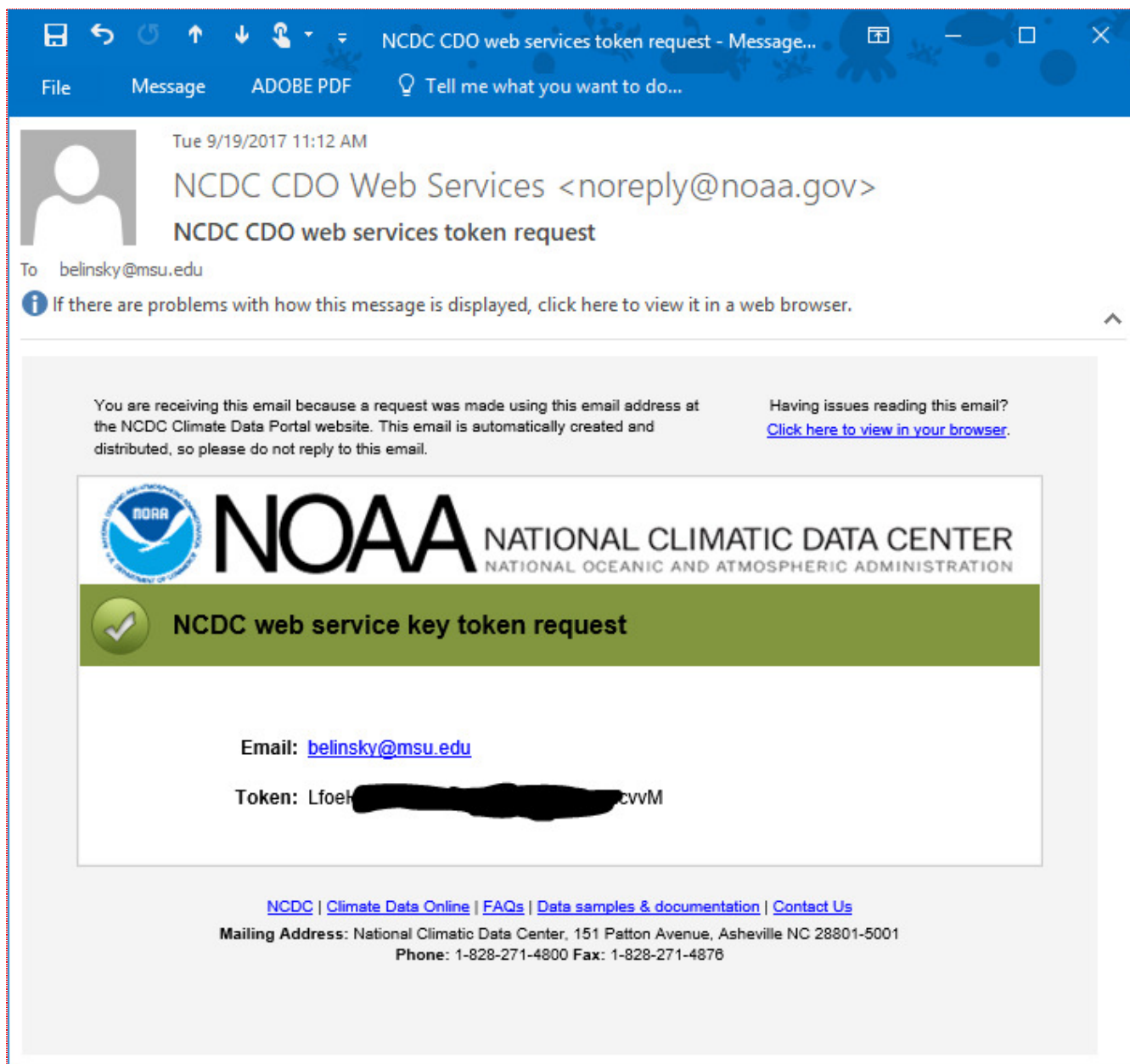


Fig 1: The email from NOAA/NCDC with your token.

3.1 - Using toolbox.r for the token

The token is used as a parameter whenever you make a call to the NOAA/NCDC database in your script. It would be nice to keep the 32 character token in a place where you can access it easily and not have to remember it. One place to keep the token is inside your **toolbox.r** script file.

Just add this line at the top of the **toolbox.r** file:

```
1 | myToken = "put your 32 character token here";
```

Now you just need to include **toolbox.r** in any script where you access the NOAA/NCDC database and you can use **myToken** to represent the 32 character token.

Let's start a new script that includes **toolbox.r**. To include a script file inside another script file, you use the **source()** function and set the parameter **file** to the location of **toolbox.r**. Note: "**scripts/toolbox.r**" assumes that you have saved **toolbox.r** inside the **scripts** folder in your **R Root** folder.

```

1 {
2   rm(list=ls()); options(show.error.locations = TRUE);
3
4   source(file="scripts/toolbox.r");
5 }

```

Because you have included **toolbox.r**, you now have access to **myToken** inside your new script file.

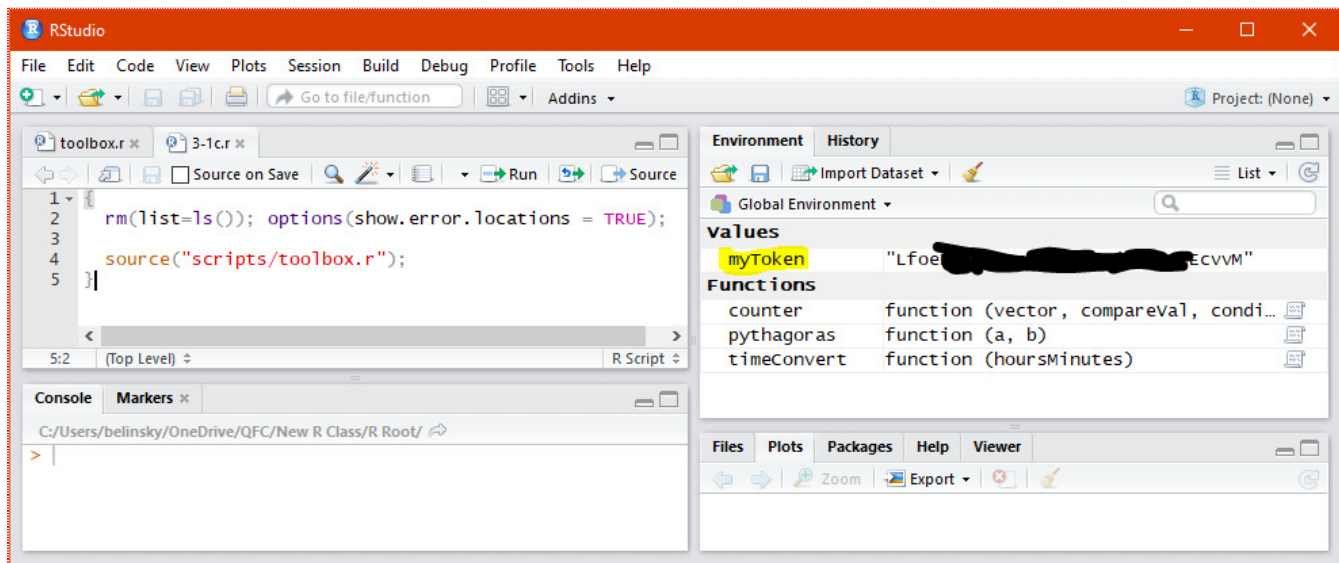


Fig 2: Including the **toolbox.r** script with the **myToken** variable.

4 - Packages (or, other peoples' functions)

In the last lesson we used some plot functions like **boxplot()** and **hist()**. These functions exist in the **R Base Package** -- the R Base Package is script that contains a large list of built-in functions. However, as we saw in the last lesson, you can create a file that contains your own functions. In fact, many people have done this and created their own public scripts that contain related functions. These public scripts are called **packages**.

In the next few lessons we will work with two packages: **rnoaa** and **reshape2**. The functions in the **rnoaa** package allow you to connect to, and gather weather information from, the NOAA/NCDC online weather database. The functions in the **reshape2** package allow you to manipulate (reshape) your data frame.

To use the functions inside the **rnoaa** and **reshape2** packages, you need to:

- 1) install **rnoaa** and **reshape2** in R
- 2) include **rnoaa** and **reshape2** in your script

4.1 - Installing the rnoaa and reshape2 package

To install the **rnoaa** package in R

- 1) click on **Tools -> Install Packages...**
- 2) type in **rnoaa** in the Packages textbox
- 3) click on **Install** (make sure "Install Dependencies" is checked)

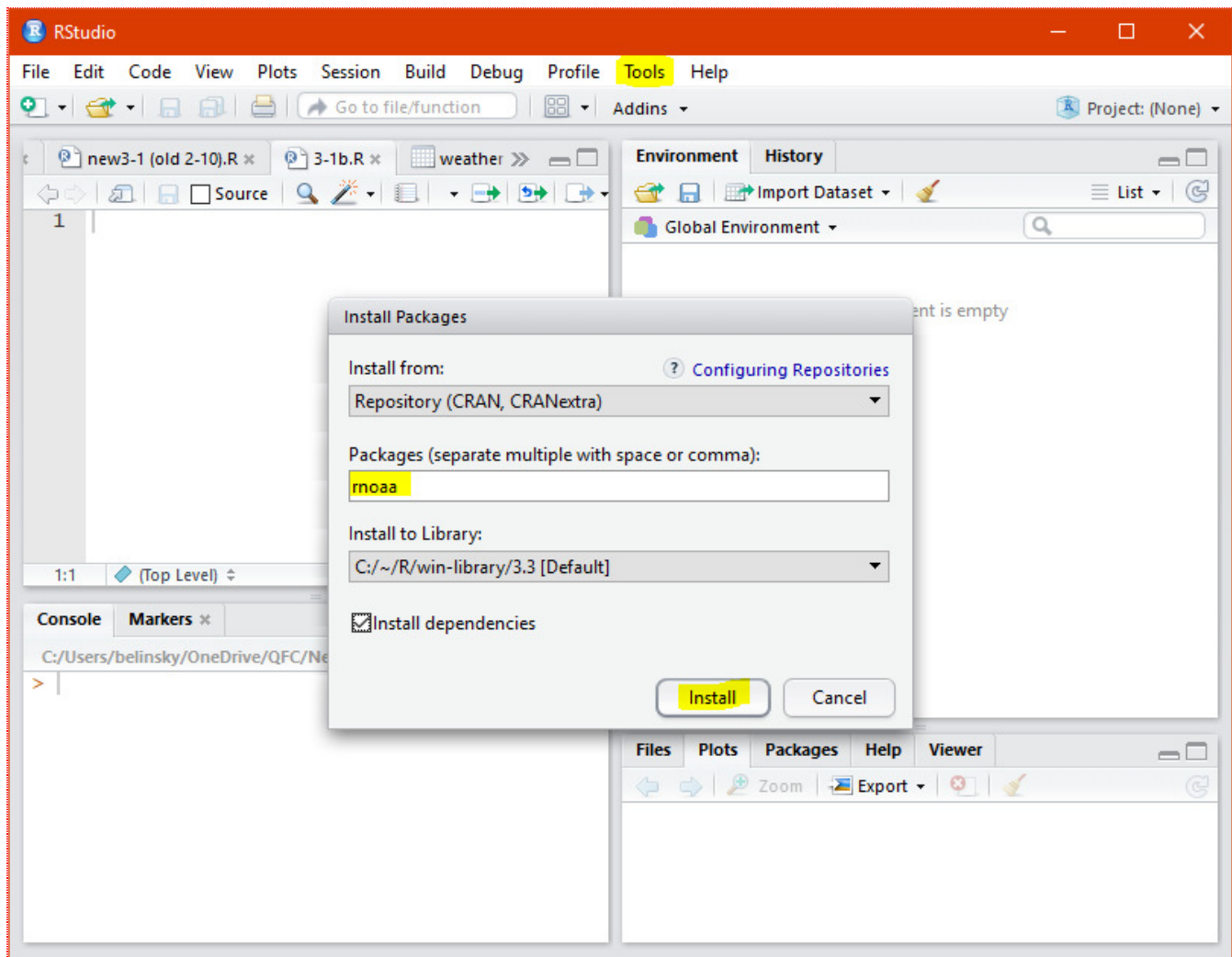


Fig 3: Installing the **rnoaa** package in R

Repeat the above steps for the **reshape2** package. These steps will work for any package you want to install in the future.

Extension: Dependencies

4.2 - Including the packages in your script

You still need to tell your script that you are going to use the functions in the **rnoaa** and **reshape2** package, which is done using the **library()** function with the parameter **package**.

```

1 {
2   rm(list=ls()); options(show.error.locations = TRUE);
3
4   source(file="script/toolbox.r");
5   library(package="rnoaa");    # include functions from the rnoaa package
6   library(package="reshape2"); # include functions from the reshape2 package
7 }

```

Now your script has access to all of the **rnoaa** and **reshape2** functions -- and, of course, your **toolbox.r**

script, which has *myToken*.

5 - The *rnoaa* package

A huge disclaimer: the *rnoaa* package espouses the problems that exists in many R packages, which includes poor documentation and unintuitive function naming.

The purpose of this lesson is not to give a complete treatise on how to use packages -- it is to give you a starting point and make using packages a little less intimidating.

The starting point for finding most R packages is [The Comprehensive R Archive Network \(CRAN\) website](#). This website holds pretty much everything R related and you will often see it on R package-related internet searches.

When I do a search for "r rnoaa" in google I get:

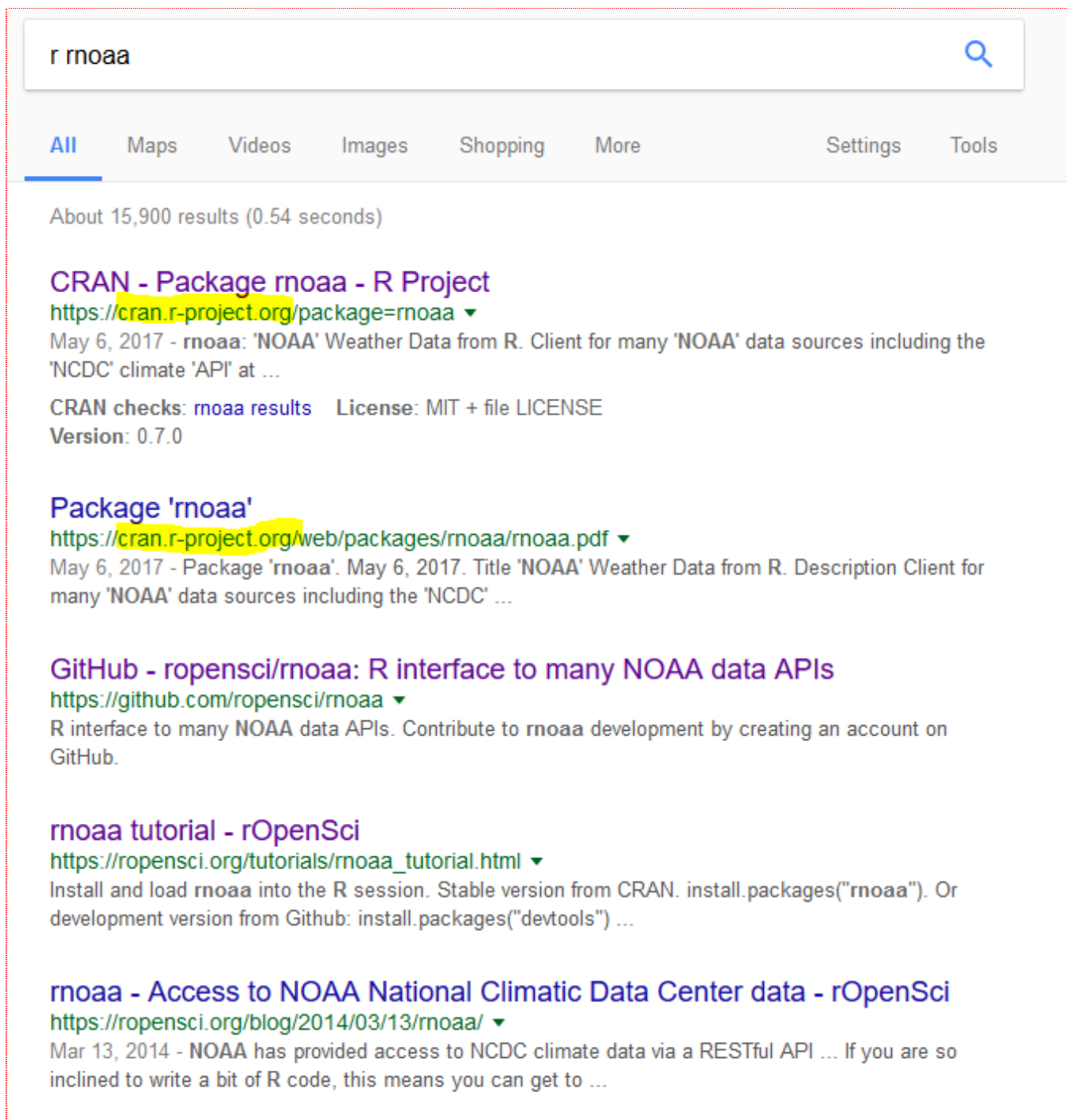


Fig 4: A google search for "r noaa" -- with results from the CRAN website.

The first two results are from the CRAN website (cran.r-project.org) and the next three are linked inside the CRAN website. So, when you see CRAN or cran.r-project.org, you probably have a good starting point for learning about the package.

5.1 - Package home page

For every package homepage inside CRAN, there are two links that are important: the **URL** and the **Reference Manual**. The URL takes you to the main page for the package and the reference manual is a pdf that lists every function inside the package.

The screenshot shows a web browser window with the CRAN package page for 'rnoaa'. The browser tabs include '02-03 Sequences', 'Edit HTML File - Te', 'R: The R Base Pack', and 'CRAN - Package r'. The address bar shows the URL 'https://cran.r-project.org/web/packages/rnoaa/'. The page content is as follows:

rnoaa: 'NOAA' Weather Data from R

Client for many 'NOAA' data sources including the 'NCDC' climate 'API' at <https://www.ncdc.noaa.gov/cdo-web/webservices/v2>, with functions for each of the 'API' endpoints: data, data categories, data sets, data types, locations, location categories, and stations. In addition, we have an interface for 'NOAA' sea ice data, the 'NOAA' severe weather inventory, 'NOAA' Historical Observing 'Metadata' Repository ('HOMR') data, 'NOAA' storm data via 'IBTrACS', tornado data via the 'NOAA' storm prediction center, and more.

Version: 0.7.0

Imports: [utils](#), [httr](#) (≥ 1.0.0), [crul](#) (≥ 0.2.0), [lubridate](#), [dplyr](#), [tidyr](#), [ggplot2](#), [scales](#), [XML](#), [xml2](#), [jsonlite](#), [rappdirs](#), [gridExtra](#), [tibble](#), [isdpaser](#) (≥ 0.2.0), [geonames](#), [hoardr](#)

Suggests: [roxygen2](#) (≥ 6.0.1), [testthat](#), [knitr](#), [taxize](#), [ncdf4](#), [leaflet](#), [rgdal](#), [rmarkdown](#), [purrr](#), [ggmap](#), [ropenag](#)

Published: 2017-05-06

Author: Scott Chamberlain [aut, cre], Brooke Anderson [ctb], Maëlle Salmon [ctb], Adam Erickson [ctb], Nicholas Potter [ctb], Joseph Stachelek [ctb], Alex Simmons [ctb], Karthik Ram [ctb], Hart Edmund [ctb]

Maintainer: Scott Chamberlain <myrmecocystus at gmail.com>

BugReports: <https://github.com/ropensci/rnoaa/issues>

License: MIT + file LICENSE

URL: <https://github.com/ropensci/rnoaa>

NeedsCompilation: no

Materials: [README](#) [NEWS](#)

CRAN checks: [rnoaa results](#)

Downloads:

Reference manual: [rnoaa.pdf](#)

Vignettes: [buoy vignette](#), [HOMR metadata](#), [NOAA NCDC dataset attributes](#), [ncdc vignette](#), [ncdc workflow](#), [air quality and weather data](#), [Sea ice vignette](#), [NOAA storms](#), [SWDI vignette](#)

Fig 5: The homepage for the *rnoaa* package.

The *rnoaa* pdf lists way more functions than you will ever use. Most of the functions listed are *helper functions*, or functions used by other functions inside the package. Helper function are rarely directly called by the user but the documentation does not differentiate the helper functions from the main functions.

R topics documented:

rnoaa-package	3
arc2	4
argo	5
autoplot.meteo_coverage	9
buoy	10
caching	12
coops	12
cpc_prdp	15
deg2rad	16
ersst	17
fipscodes	18
gefs	19
ghcnd	21
ghcnd_search	23
ghcnd_splitvars	24
ghcnd_states	25
ghcnd_stations	26
homr	27
homr_definitions	29
isd	30
isd_read	32
isd_stations	34
isd_stations_search	35
meteo_clear_cache	37
meteo_coverage	37
meteo_distance	38
meteo_nearby_stations	40
meteo_process_geographic_data	42
meteo_pull_monitors	43
meteo_show_cache	46
meteo_spherical_distance	46
meteo_tidy_ghcnd	47
meteo_tidy_ghcnd_element	49
ncdc	50
ncdc_combine	54
ncdc_datacats	56
ncdc_datasets	58
ncdc_datatypes	60
ncdc_locs	62
ncdc_locs_cats	64
ncdc_plot	66

Fig 6: A partial list of functions on page 2 of the *rnoaa* pdf.

Finding out which functions to use can be tricky but if you go through the community page at the **URL** you will notice that the function **ncdc()** is often used to get weather data. In fact **ncdc()** is meant to be the broad function that links to the NOAA/NCDC databases and this is the function we will use.

6 - Getting data using the *rnoaa* package

We want to get all of the weather information for 2016 from Lansing, Michigan. To do this we will use the **ncdc()** function in the **weatherData** package.

ncdc() looks like this (from page 9 on the [rnoaa.pdf](#)):


```
1 |   ncdc(datasetid = NULL, datatypeid = NULL, stationid = NULL,  
2 |       locationid = NULL, startdate = NULL, enddate = NULL, sortfield = NULL,  
3 |       sortorder = NULL, limit = 25, offset = NULL, token = NULL,  
4 |       dataset = NULL, datatype = NULL, station = NULL, location = NULL,  
5 |       locationtype = NULL, page = NULL, year = NULL, month = NULL,  
6 |       day = NULL, includemetadata = TRUE, results = NULL, ...)
```

The (...) at the end of **ncdc()** means there are many more optional parameters to change but we have no need to change the default values on any of these parameters.

The parameters we will be changing in **ncdc()** are (i.e., the parameters whose values we will change from the default value):

- **datasetid**: the database we want the data from
- **datatypeid**: the data we want from the database
- **stationid**: the weather station we are getting data from
- **startdate**: start date for the data
- **enddate**: end date for the data
- **token**: the 32 character token you got from NOAA/NCDC
- **limit**: the maximum number of data points that will be returned to you

6.1 - Using your token

We have **toolbox.r** included in our script and **toolbox.r** has a variable named **myToken** that was assigned the 32 character token you got from NOAA/NCDC. So we can use **myToken** as the token:

```
1 |   token = myToken
```

6.2 - The limit parameter

The **limit** parameter is the maximum amount of data you will get. This is mostly to protect NOAA/NCDC from people accidentally (or maliciously) requesting millions of data points and crashing the server! The highest number you can put is 1000. You generally want to put in a number that is a little higher than the amount of data you expect to receive. We are getting 10 data points for every day for one month so we are getting around 300 data points. To be safe, we will set the limit to 400.

```
1 |   limit = 400
```

Note: If you do not receive all the data from NOAA/NCDC that you requested, there is a good chance that NOAA/NCDC cut off the data because there were too many data points. It is easy to miscalculate the number of data points, so it is always good idea to check the data you received for completeness.

6.3 - The start and end date parameters

Alright, let's get to the real data. **startdate** and **enddate** are just like they sound -- they give the date range for the data.

We want all of January 2016 so:

```
1 | startdate = "2016-01-01", enddate = "2016-01-31"
```

note the YEAR-MONTH-DAY format for the dates.

6.4 - The datasetid parameter

datasetid (*data set id*) is the name of the database you want to get data from.

We are going to using the Global Historical Climatology Network-Daily database -- the id is **GHCND**.

```
1 | datasetid = "GHCND"
```

Extension: Finding the datasetid

6.5 - The stationid parameter

The **stationid** is a unique string for a weather station that is 17 characters long and starts with "GHCND:". [You can find station ids on this page](#). Just type in a search parameter and click "Search".

For this lesson we will use the Lansing Capital City Airport station -- id: **GHCND:USW00014836**

```
1 | stationid = "GHCND:USW00014836"
```



The screenshot shows the NOAA website's search results page. The search bar at the top left contains the word "lansing". Below the search bar, a list of search results is displayed. The results include station names, their IDs, and periods of record. The last result, "LANSING CAPITAL CITY AIRPORT, MI US", is highlighted with a yellow box, showing its station ID as "GHCND:USW00014836". To the right of the search results is a map of the United States with a blue pin indicating the location of Lansing, Michigan. The map shows major cities and state boundaries.

Fig 7: Searching for a station id -- in this case we will use the one at Lansing Airport

6.6 - the datatypeid

Finally, we need to tell NOAA/NCDC what kind of data we want (e.g., temperature, rainfall, etc).

If you go to the [Lansing Capital City Airport station page](#) and scroll to the bottom, you will see a section called **Available Data Type**. Clicking on the categories produces a list of data types and their 4 digit code. It is these 4 digits codes that are used for the **datatypeid**.


NOAA NATIONAL CENTERS FOR ENVIRONMENTAL INFORMATION
NATIONAL OCEANIC AND ATMOSPHERIC ADMINISTRATION


[Home](#) [Climate Information](#) [Data Access](#) [Customer Support](#) [Contact](#) [About](#)

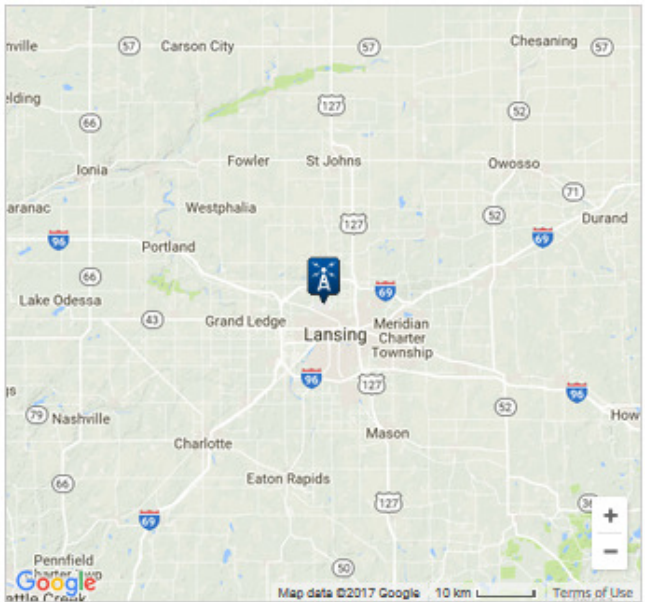
[Home > Climate Data Online > Station Details](#)
[Datasets](#) | [Search Tool](#) | [Mapping Tool](#) | [Data Tools](#) | [Help](#)

Daily Summaries Station Details

STATION DETAILS	
Name	LANSING CAPITAL CITY AIRPORT, MI US
NetworkID	GHCND:USW00014836
Latitude/Longitude	42.78028°, -84.57889°
Elevation	256.3 m

PERIOD OF RECORD	
Start Date¹	1948-01-01
End Date¹	2017-10-20
Data Coverage²	93%

ADD TO CART



Station Data Inventory, Access & History

Data & Inventory
[View Data](#)
Available Data Types
[Air Temperature](#)
[Precipitation](#)
[Sky cover & clouds](#)
[Sunshine](#)
[Water](#)
[Weather Type](#)
[Wind](#)

DATA TYPE	DESCRIPTION	START	END	COVERAGE % ²
PRCP	Precipitation	1948-01-01	2017-10-18	93%
SNOW	Snowfall	1948-01-01	2017-10-18	84%
SNWD	Snow depth	1948-01-01	2017-10-18	85%

1-3 of 3

Fig 8: Location of the 4-digit **datatypeids**.

For this example, we will use the following codes:

TMAX, TAVG, TMIN, PRCP, SNOW, AWND, WDF2, WSF2, WT01, WT09

We need to put these IDs in a vector to assign them to the parameter **datatypeid**

```

1 datatypeid=c("TMAX", "TAVG", "TMIN", "PRCP", "SNOW", "AWND",
2             "WDF2", "WSF2", "WT01", "WT09")

```

Note: if you are really bored and want to know everything about the data contained with the GHCND data set, [you can consult this document](#).

6.7 - Putting the function together

Here is the full function call to NOAA/NCDC with all the parameters. I assign the return value from ***ncdc()*** to the variable ***lansingWeather***

```
1 {
2   rm(list=ls()); options(show.error.locations = TRUE);
3
4   source("scripts/toolbox.r");
5   library("rnoaa"); # include functions from the rnoaa package
6   library("reshape2"); # include functions from the reshape2 package
7
8   lansingweather = ncdc(datasetid='GHCND',
9                        datatypeid=c("TMAX", "TAVG", "TMIN", "PRCP",
10                                     "SNOW", "AWND", "WDF2", "WSF2",
11                                     "WT01", "WT02", "WT03"),
12                        stationid='GHCND:USW00014836',
13                        startdate = '2016-01-01', enddate = '2016-01-31',
14                        token=myToken,
15                        limit=400 );
16 }
```

ncdc() ***lansingWeather*** is a list. You can think of lists as folders that contain objects like vectors, data frames, matrices, or even other lists (e.g., folder inside a folder). In this case, the list ***weatherData*** contains two objects: ***meta***, another list, and ***data***, a data frame. ***data*** contains the weather information we requested.

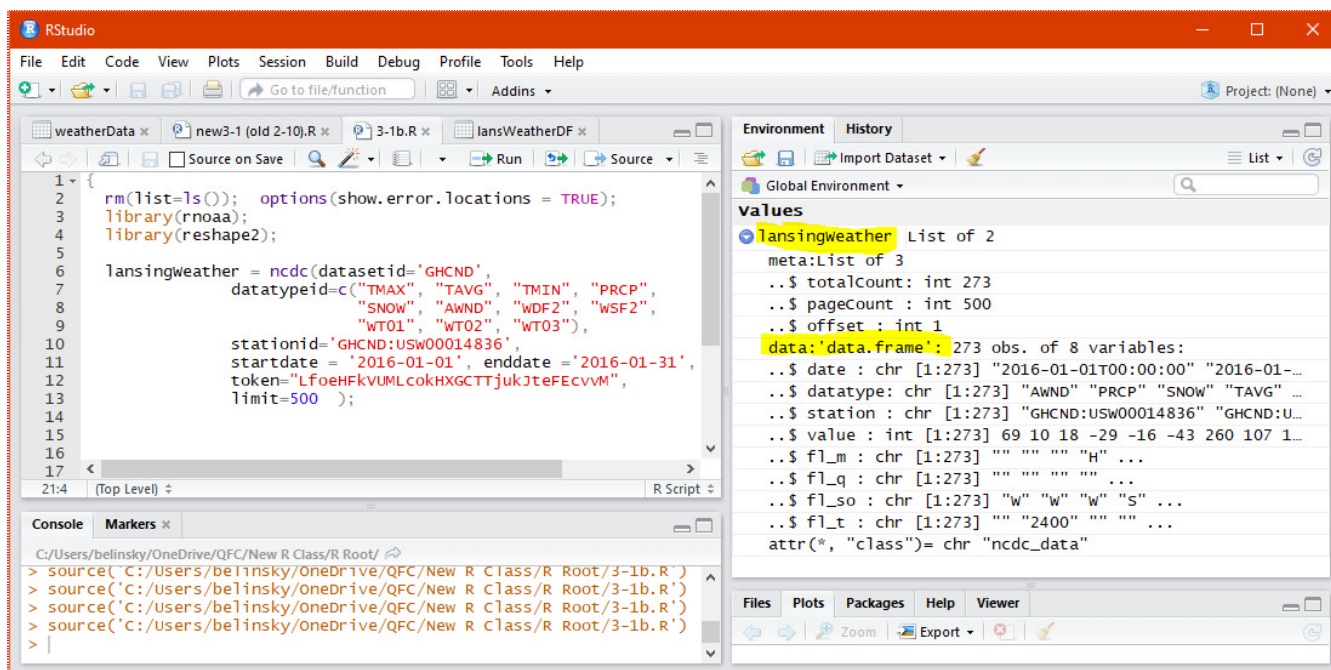


Fig 9: **list** object return from the function call to the NOAA/NCDC database.

Note: We are making requests to the NOAA/NCDC servers and these servers are not the fastest, so it might take some time for your script to run.

6.8 - Subsetting a list

We want to extract the data frame **data** from the **list** object named **lansingWeather**. Essentially we want to pull an object out of **lansingWeather**. To pull an object out of a list, we use double brackets `[[]]`. Note: to subset a data frame we used single brackets `[]`.

The following line gets the data frame named **data** from the **lansingWeather** list and saves it to the variable **lansingWeatherDF**.

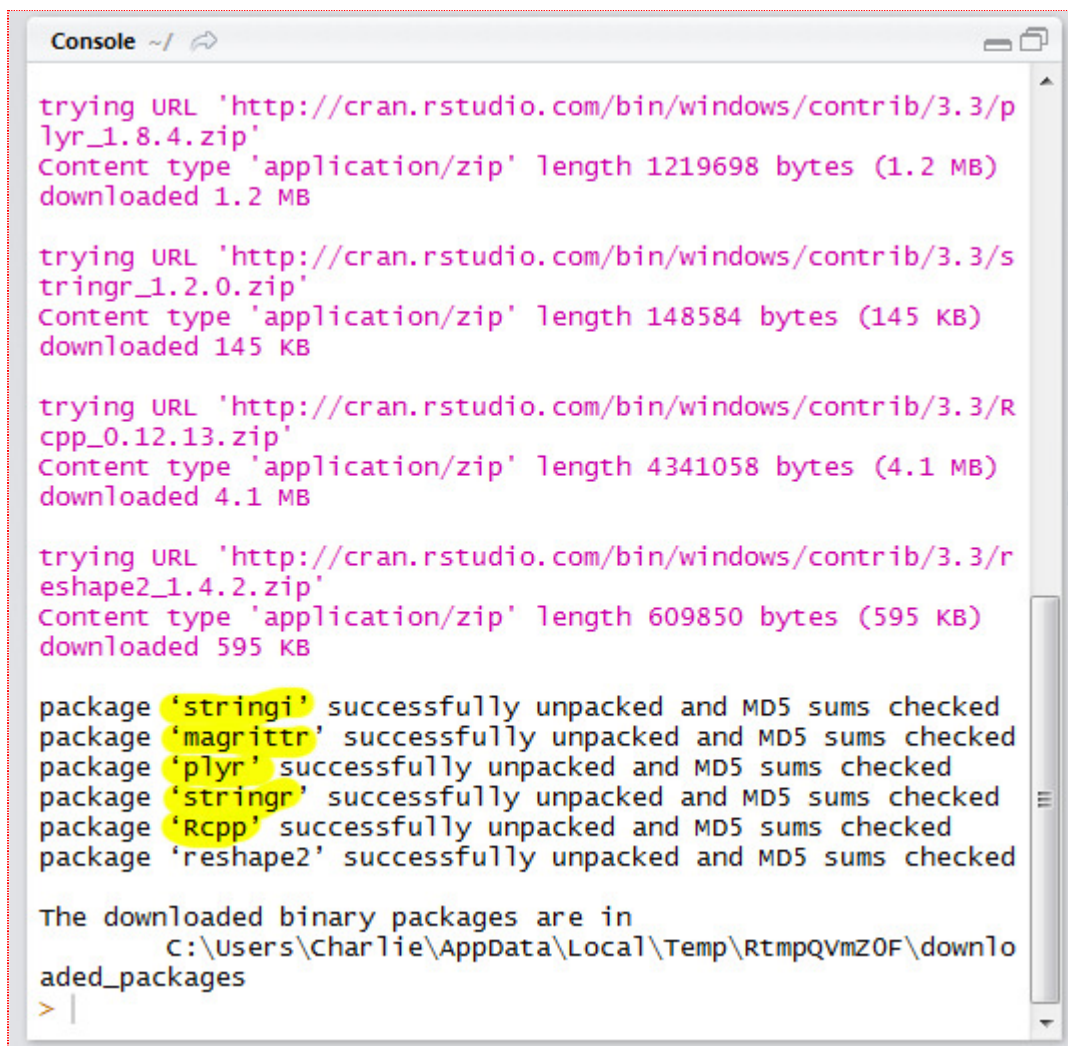
```
1 lansingWeatherDF = lansingWeather[["data"]];
```



7 - Application

- 1) Get the following data for December 2016 through February 2017 in Lansing, MI: **PRCP, SNOW, TAVG, TMAX, TMIN**
- 2) Save the data frame (**data**) from the list to a variable named **lansingWinterWeather**

8 - Extension: Dependencies

A package is just somebody else's script that you are including in your own script file. These packages are almost always dependent upon other scripts, or packages. These are its **dependencies** and the dependencies get installed with the package. That is why you might see something like this when installing a package:



```
Console ~/  

trying URL 'http://cran.rstudio.com/bin/windows/contrib/3.3/p
lyr_1.8.4.zip'
Content type 'application/zip' length 1219698 bytes (1.2 MB)
downloaded 1.2 MB

trying URL 'http://cran.rstudio.com/bin/windows/contrib/3.3/s
tringr_1.2.0.zip'
Content type 'application/zip' length 148584 bytes (145 KB)
downloaded 145 KB

trying URL 'http://cran.rstudio.com/bin/windows/contrib/3.3/r
cpp_0.12.13.zip'
Content type 'application/zip' length 4341058 bytes (4.1 MB)
downloaded 4.1 MB

trying URL 'http://cran.rstudio.com/bin/windows/contrib/3.3/r
eshape2_1.4.2.zip'
Content type 'application/zip' length 609850 bytes (595 KB)
downloaded 595 KB

package 'stringi' successfully unpacked and MD5 sums checked
package 'magrittr' successfully unpacked and MD5 sums checked
package 'plyr' successfully unpacked and MD5 sums checked
package 'stringr' successfully unpacked and MD5 sums checked
package 'Rcpp' successfully unpacked and MD5 sums checked
package 'reshape2' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
  C:\Users\Charlie\AppData\Local\Temp\RtmpQVmZ0F\downlo
aded_packages
> |
```

Fig 10: Installing **reshape2** package, which has 5 package dependencies.

Reshape2 is dependent upon 5 other packages and RStudio has installed those packages for you, too (because you had "Install Dependencies" checked).

9 - Extension: Finding the **datasetid**

Unfortunately, the help pages for the NOAA/NCDC databases are quite lacking, so finding what the names of the databases (i.e., the **datasetid**) is not elementary.

[This webpage](#) presents the different data sets but does not give you the ids. I still have not found good documentation about this, however, **GHCND** is probably the most common **datasetid**.