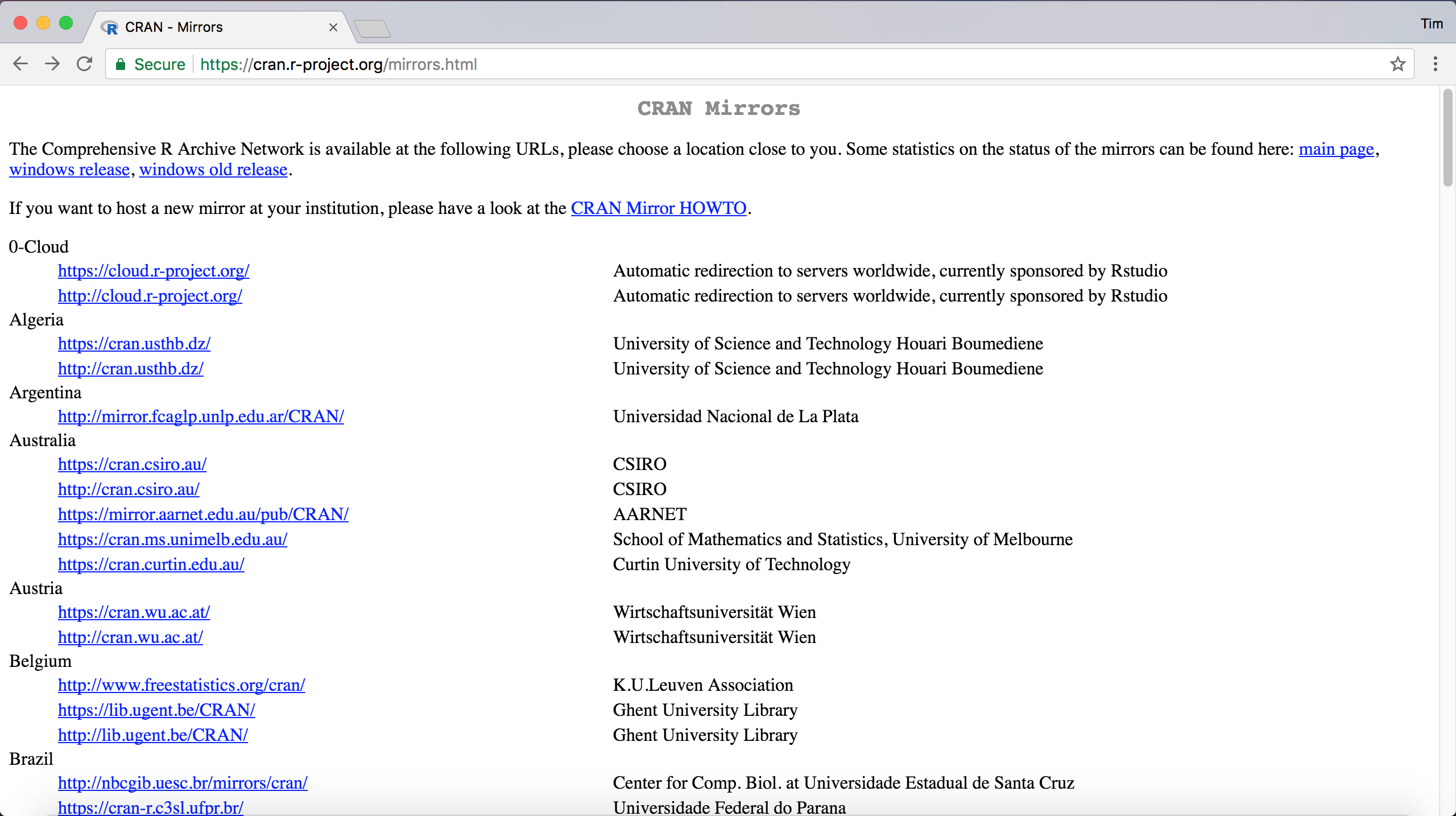
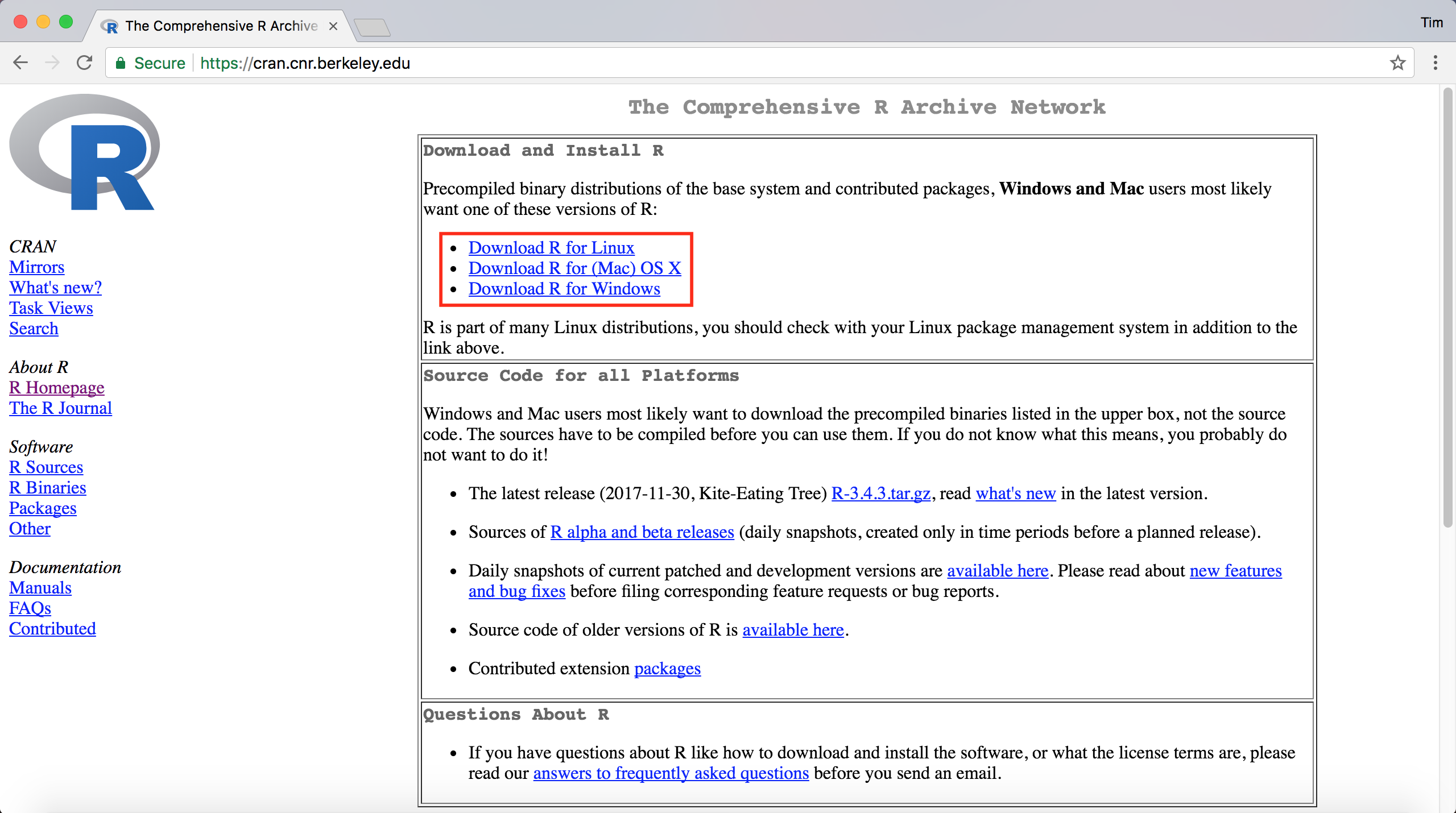
**CGM Analysis New-User Guide**

Installing R for the first time

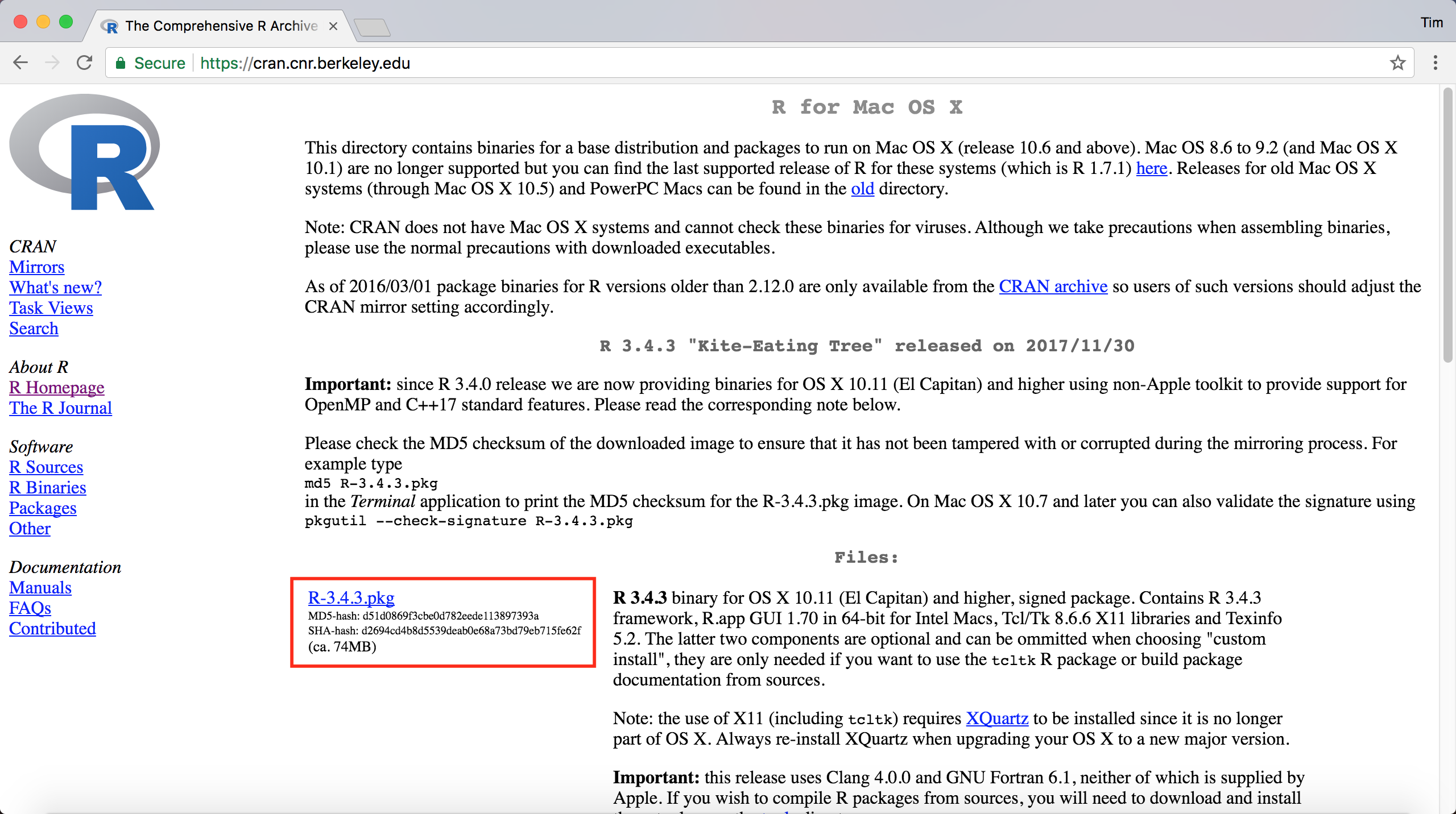
1. To download R, go to <https://cran.r-project.org/mirrors.html> and choose your preferred download location (for this example we use the UC Berkeley mirror).



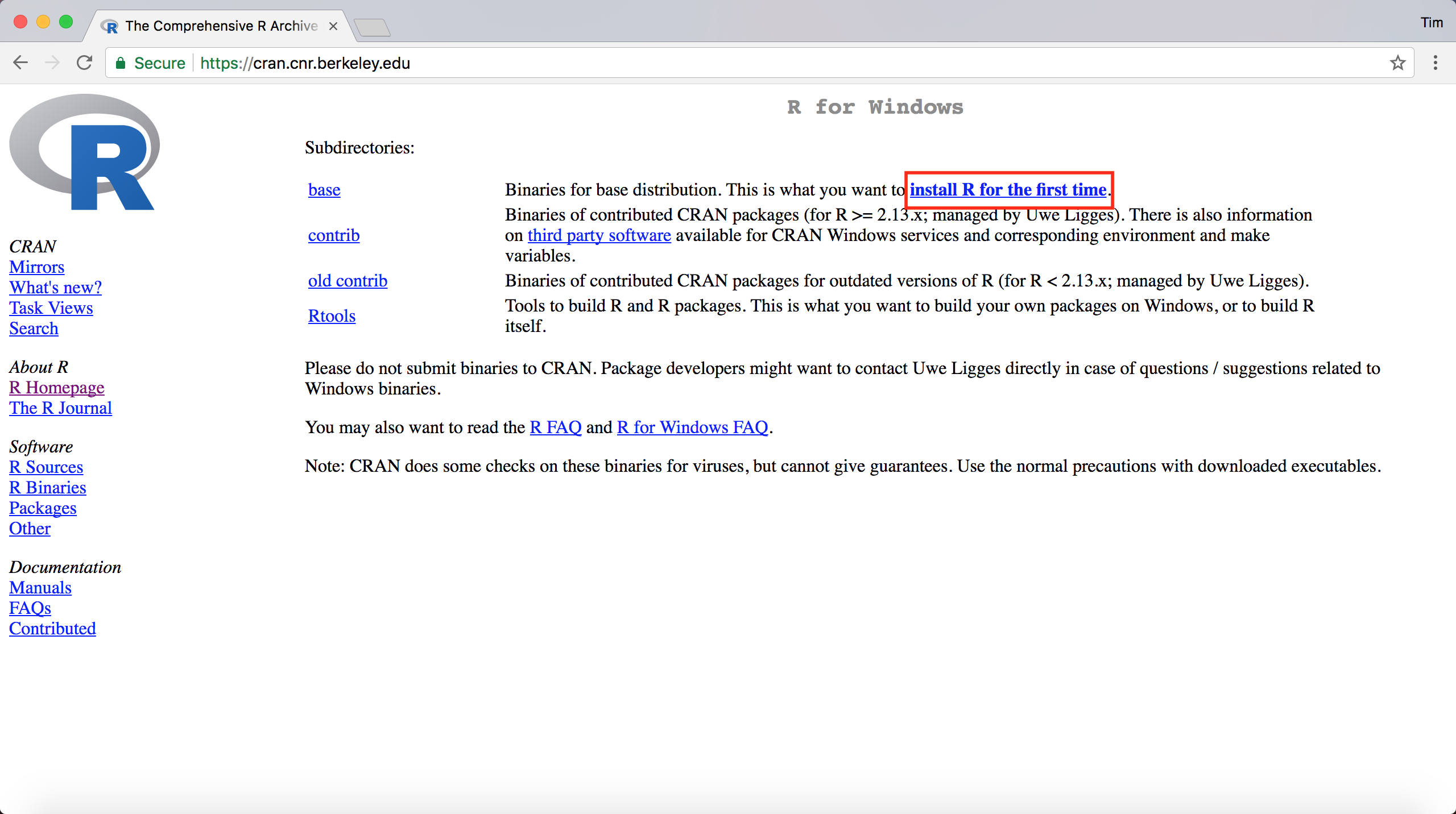
1. Click on the download link for your operating system. Mac users will then click directly on the package download link, but Windows users will need to click “install R for the first time” to get to the download page.

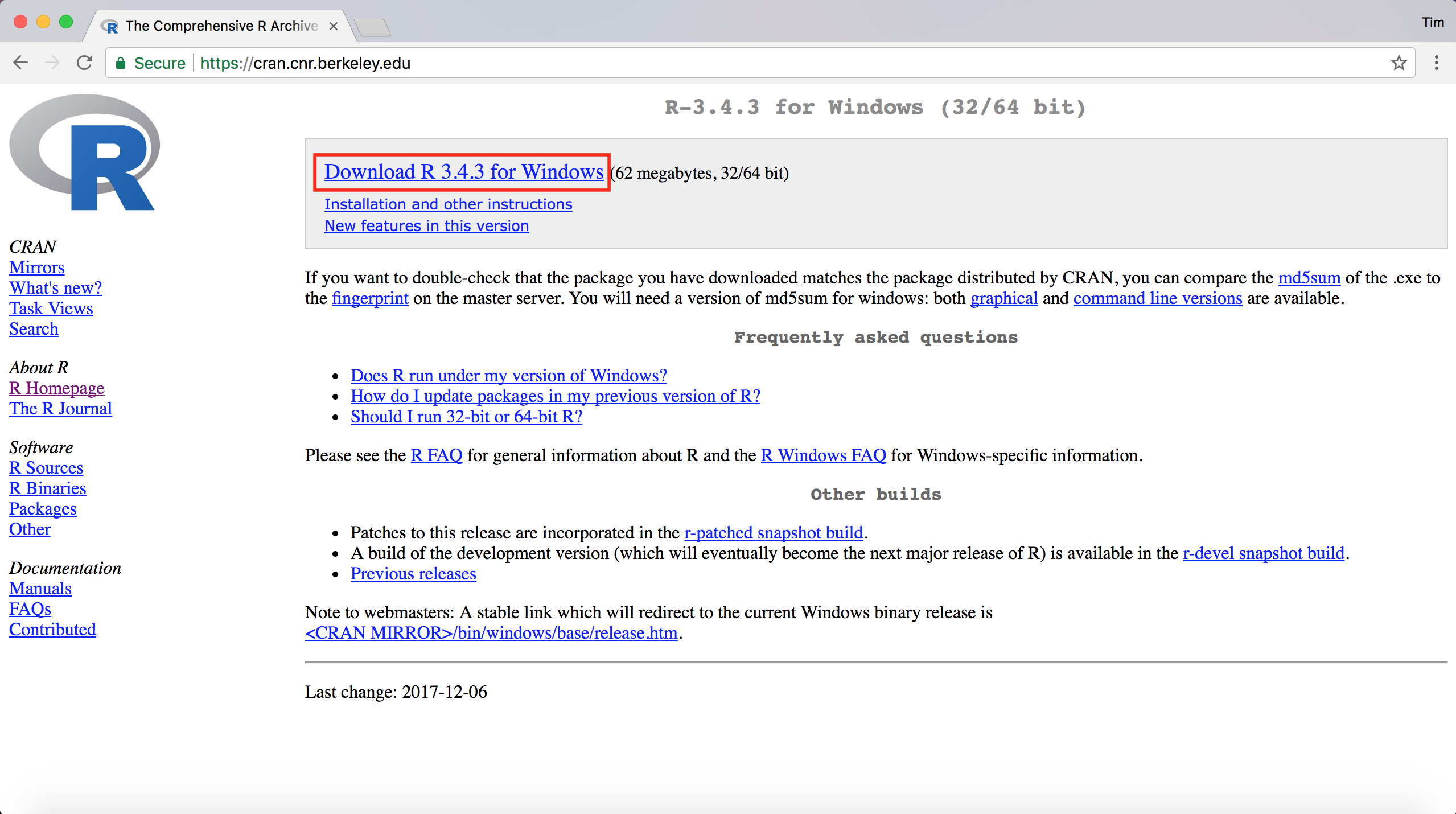


Mac:



Windows:





1. Linux Ubuntu users will first need to add

“deb https://<my.favorite.cran.mirror>/bin/linux/ubuntu xenial” (without quotes) to /etc/apt/sources.list, and then can simply install R and RStudio from the command line using:

sudo apt-get update

sudo apt-get install r-base

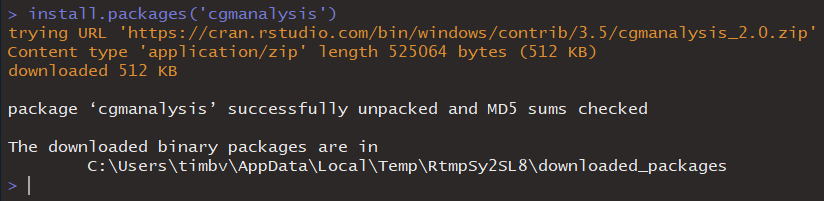
sudo apt-get install rstudio

1. Open the installer download, and follow the prompts for a standard installation of R. Although the standard R installation includes its own graphical user interface (GUI), we highly recommend using RStudio instead, and **the rest of this tutorial’s examples will be shown using RStudio**.

Please visit <https://www.rstudio.com/products/rstudio/download/#download> to download and install RStudio for your operating system.

Installing the package from CRAN

1. After installing R and the GUI of your choice, type: install.packages('cgmanalysis') and hit enter.



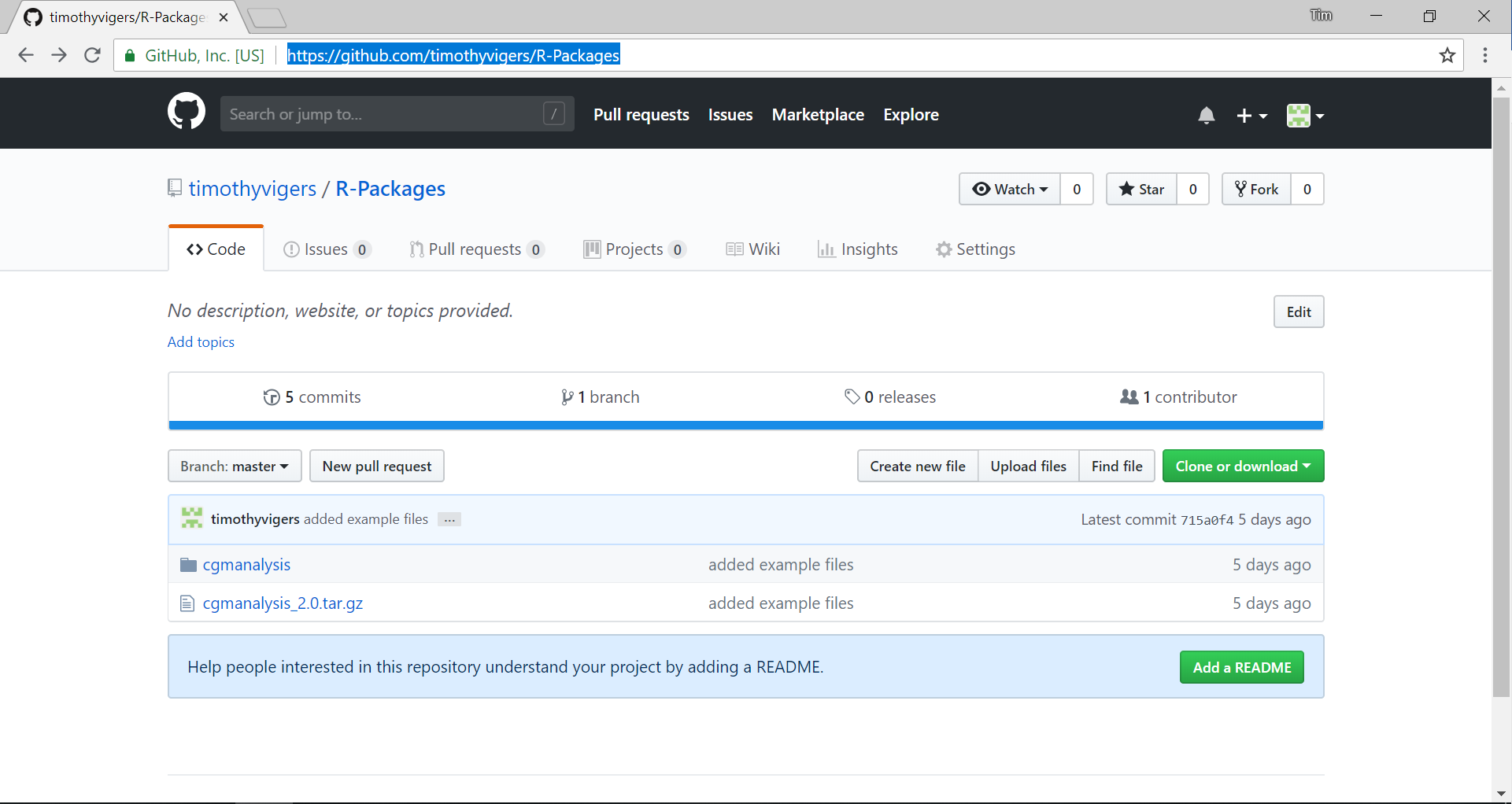
1. Once the package is installed, you will need to load it by typing: library(cgmanalysis).



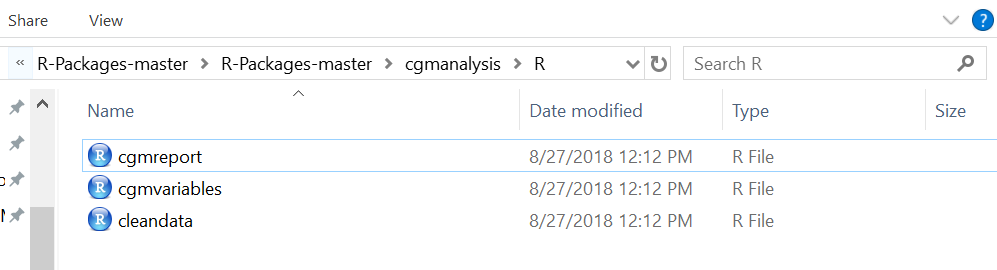
1. Once the package is installed and loaded into R, you can use the functions as described below in the “Using the R functions” section.

Downloading the code from GitHub

1. Go to the CGM analysis GitHub repository (at <https://github.com/timothyvigers/R-Packages> if you did not already download the user guide from this location). Click “Clone or download” and then select “Download ZIP.” You can also clone the repository if you are already familiar with GitHub, but using repositories is beyond the scope of this guide.



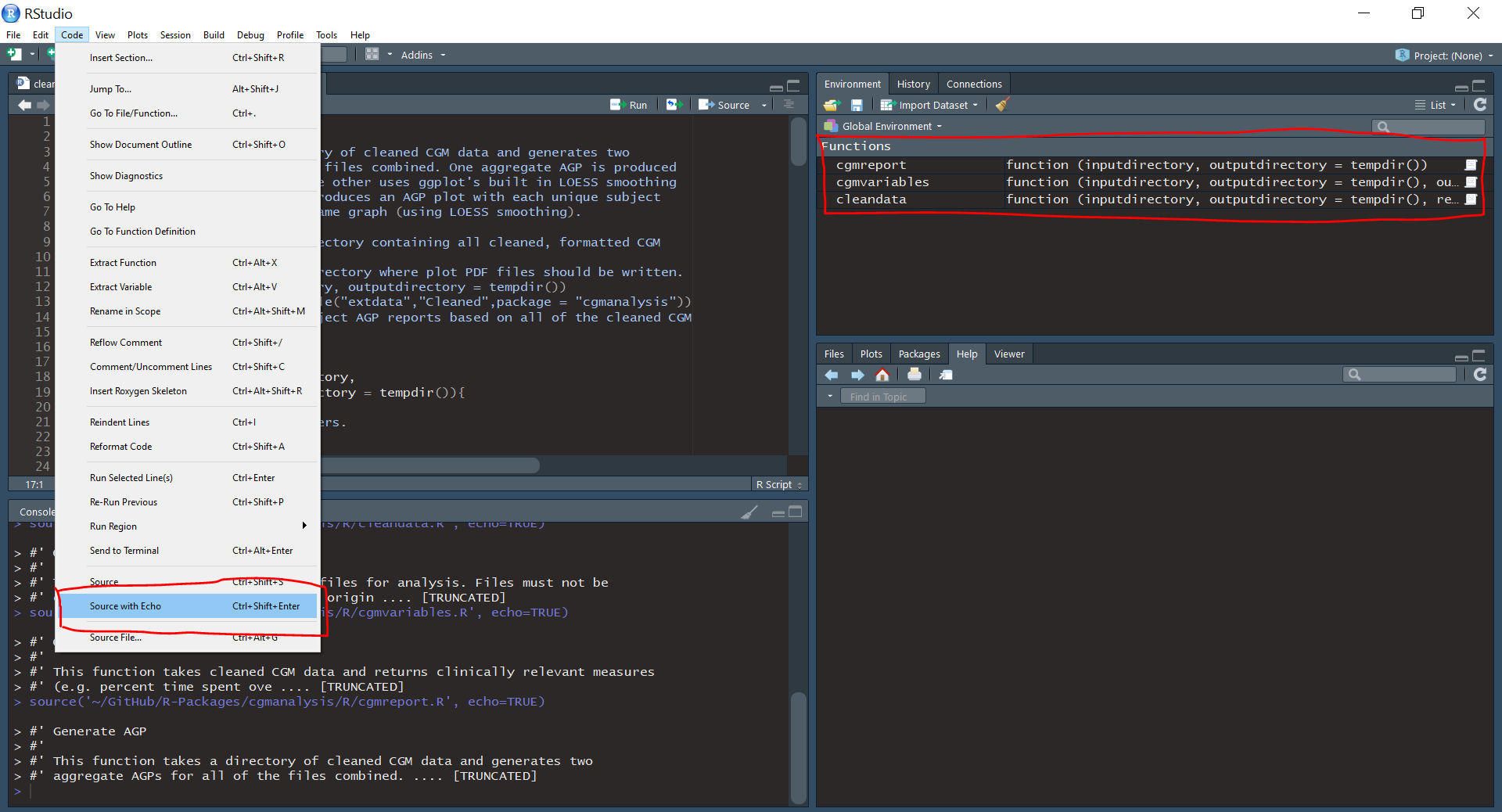
1. Extract the .zip file and check that it contains the same files as the repository. Note that the contents may look different from the screenshot above, as we change and add to the program.
2. Move the contents of the folder to your desired location, so they can be used again in the future.
3. Open “Downloads\R-Packages-master\R-Packages-master\cgmanalysis\R” to see the functions included in this package. Open the functions with your preferred R GUI.



Using the R functions

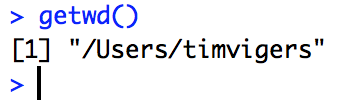
**Note: the following instructions are for those using RStudio, and the steps may look slightly different for anyone using another R GUI. Also, the following screenshots are taken from a machine running Windows 10, and certain things will look different for Mac or Linux users.**

1. If you downloaded the code from GitHub, then for each function file, click “Code” then “Source with Echo.” This will load the functions into the R environment, and you should see them populate in the top right corner.



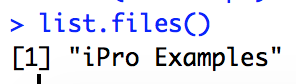
This step is not necessary if you downloaded the package from CRAN and loaded it into R using library(cgmanalysis), as in the instructions above.

1. Find the directory you are currently working in by typing: getwd.



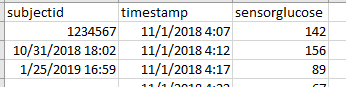
1. If you are not in the correct directory (i.e. the folder above the directory where the CGM data are stored), you can change the working directory by typing:

../../../Desktop/Screen%20Shot%202017-12-07%20at%201.02.23%20PM.p

1. To ensure that you have set the working directory correctly, type:

Check that the folder containing your CGM data is listed in the output of this command. For the purposes of this guide we’ll be working with the “iPro Examples” directory, which contains 5 CGM data sets exported from our iPro database.

1. If your folder of CGM files is available in the working directory, you can run the cleandata() function on the folder to get the CGM files ready for analysis. For this function to work, CGM data must be saved in the original format. Manual edits can sometimes make it difficult to recognize the type of CGM used, which will lead to data cleaning errors. If you need to manually clean the CGM data, make sure to save in the same format as the cleaned examples provided with these functions (see below). The subjectid column should contain the ID, then the CGM placement time, then the cgm removal time.



To clean the data, type and enter: cleandata(inputdirectory, outputdirectory). For example, if you want to clean the data in the folder titled “De-identified” and export everything to the current working directory, you would enter the command cleandata("De-identified", getwd()).

There are several other parameters you can change, depending on your needs, which you can select by typing any of the following arguments, separated by a comma:

cleandata(inputdirectory,

outputdirectory,

removegaps = TRUE,

gapfill = TRUE,

maximumgap = 20)

removegaps determines whether the data are cleaned or not. If set to TRUE, any gaps in the data will be removed or interpolated. The tail end of the data will also be trimmed to ensure the timeseries is in discrete 24-hour chunks.

If gapfill is set to TRUE (and if removegaps = TRUE), gaps smaller than or equal to maximumgap will be interpolated rather than removed.

maximumgap allows the user to determine the longest data gap (in minutes) that will be interpolated.

1. Once your data has been formatted correctly, you can analyze it by typing: cgmvariables(inputdirectory, outputdirectory)

This will analyze the cleaned data stored in the input folder (the output from the previous step) and will export a REDCap upload-ready table of CGM variables. You can also change where the output is stored by altering the outputdirectory argument as outlined previously.

Like the cleandata() function, there are some parameters that can be changed by the user for cgmvariables().

cgmvariables( inputdirectory,

outputdirectory,

outputname = "REDCap Upload",

aboveexcursionlength = 35,

belowexcursionlength = 10,

magedef = "1sd")

outputname just changes the name of the file containing final CGM variables (do not include the file extension in your new name).

aboveexcursionlength determines the number of minutes blood sugar must be above threshold to count as an excursion. The current default is 35 minutes.

belowexcursionlength determines the number of minutes blood sugar must be below threshold to count an excursion. The current default is 10 minutes.

magedef determines how large an excursion needs to be to count in the MAGE calculation. The options are 1 standard deviation, 1.5 standard deviations, or 2 standard deviations (“1sd”, “1.5sd”, “2sd”). The current default is 1 SD.

1. You can also use the directory of cleaned CGM data to generate an AGP style report, both for individual patients and in aggregate. This function currently only has options for changing the input and outputdirectory, but we plan to allow the user to determine which plots to make (and what to call them) in the future.

The printouts titled "AGP\_Tukey.pdf" and "Aggregate\_AGP\_Loess.pdf" are aggregate plots that combine all the data in the input directory. This is useful for visualizing the range, median, etc. for an entire group.

“AGP\_Loess\_Subject.pdf” shows each participant as a single line and uses ggplot’s built in smoothing function.