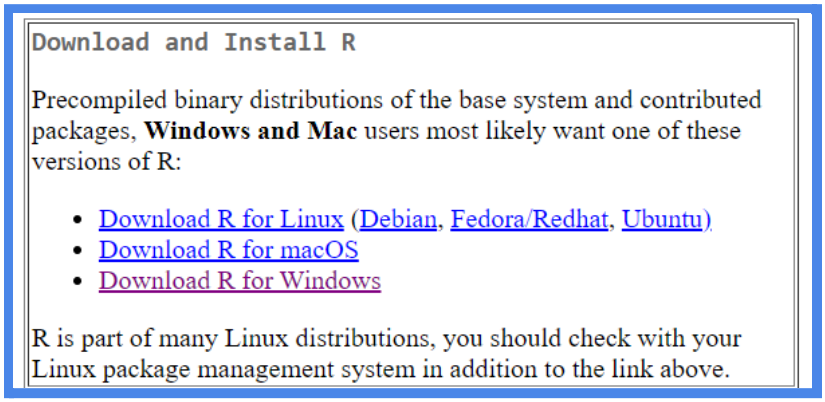
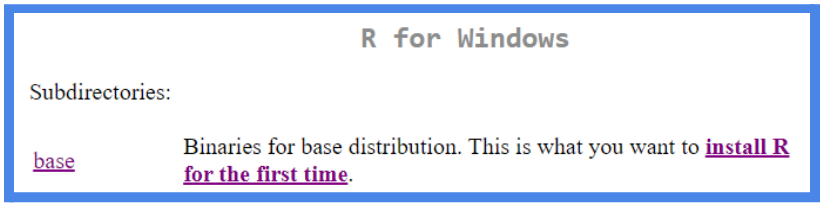
Installing R, R Tools, and RStudio on Windows

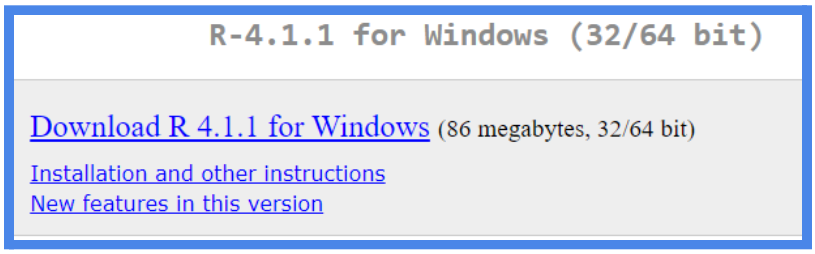
In this tutorial, we will install R, R tools, and RStudio for Windows. Links will be included in the video description.

First, we will install R. https://cran.rstudio.com/

Select Download R from Windows, then select the link for “install R for the first time. Now select Download R for Windows.

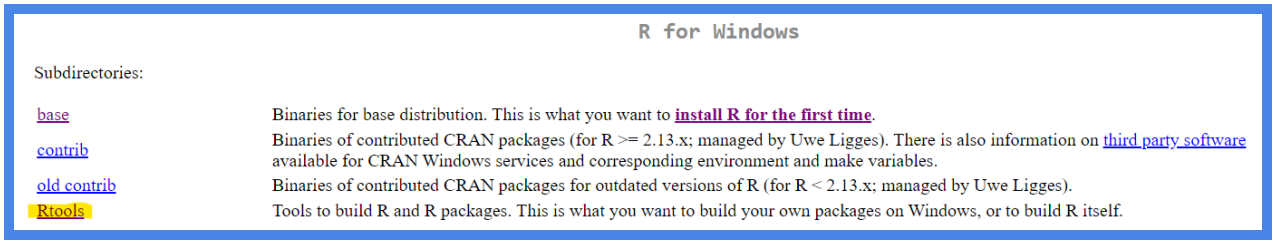






Once the file has downloaded, double click to start the installer. Click Next through all of the install prompts to install, and accept the default options.

Now we will install RTools, an external program that helps R run on Windows. Use the link below, or press the back arrow on the current page (https://cran.rstudio.com/)

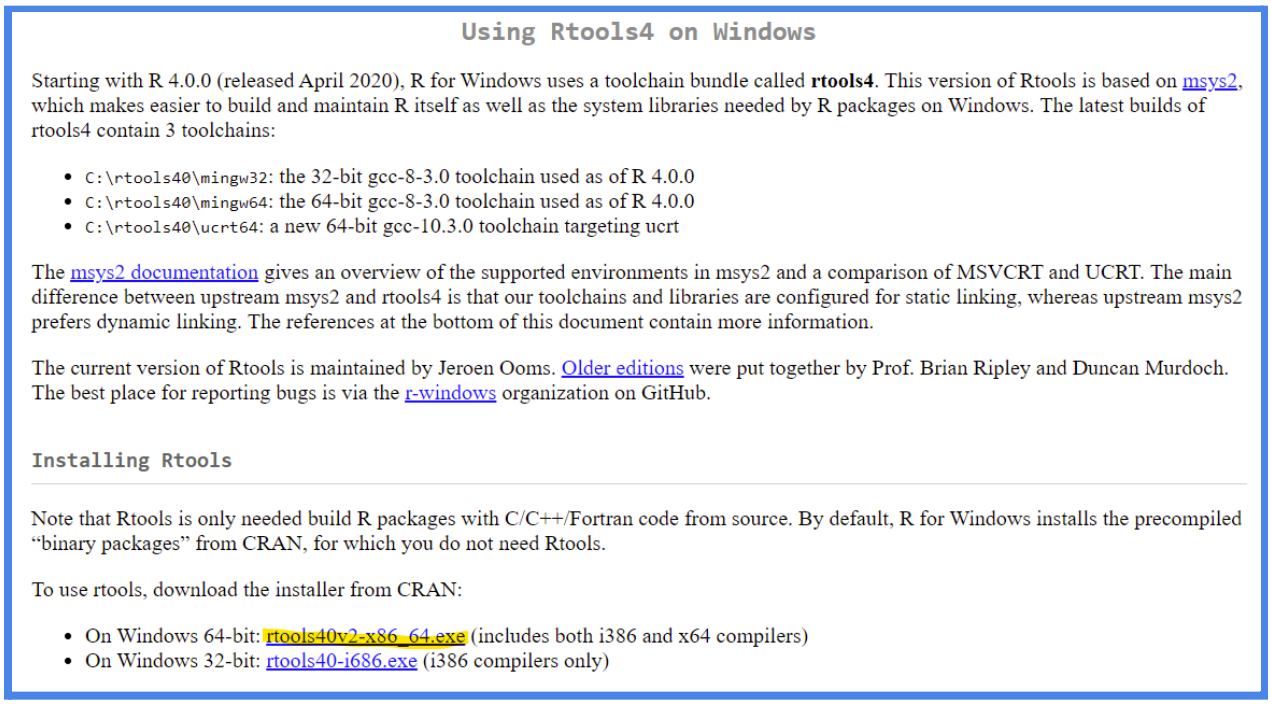


R for Windows

Subdirectories:

base	Binaries for base distribution. This is what you want to **install R for the first time**.
contrib	Binaries of contributed CRAN packages (for R >= 2.13.x; managed by Uwe Ligges). There is also information on third party software available for CRAN Windows services and corresponding environment and make variables.
old contrib	Binaries of contributed CRAN packages for outdated versions of R (for R < 2.13.x; managed by Uwe Ligges).
Rtools	Tools to build R and R packages. This is what you want to build your own packages on Windows, or to build R itself.

Most likely you will need the 64-bit version.



Using Rtools4 on Windows

Starting with R 4.0.0 (released April 2020), R for Windows uses a toolchain bundle called **rtools4**. This version of Rtools is based on msys2, which makes easier to build and maintain R itself as well as the system libraries needed by R packages on Windows. The latest builds of rtools4 contain 3 toolchains:

- `C:\rtools40\mingw32`: the 32-bit gcc-8-3.0 toolchain used as of R 4.0.0
- `C:\rtools40\mingw64`: the 64-bit gcc-8-3.0 toolchain used as of R 4.0.0
- `C:\rtools40\ucrt64`: a new 64-bit gcc-10.3.0 toolchain targeting ucrt

The msys2 documentation gives an overview of the supported environments in msys2 and a comparison of MSVCRT and UCRT. The main difference between upstream msys2 and rtools4 is that our toolchains and libraries are configured for static linking, whereas upstream msys2 prefers dynamic linking. The references at the bottom of this document contain more information.

The current version of Rtools is maintained by Jeroen Ooms. Older editions were put together by Prof. Brian Ripley and Duncan Murdoch. The best place for reporting bugs is via the r-windows organization on GitHub.

Installing Rtools

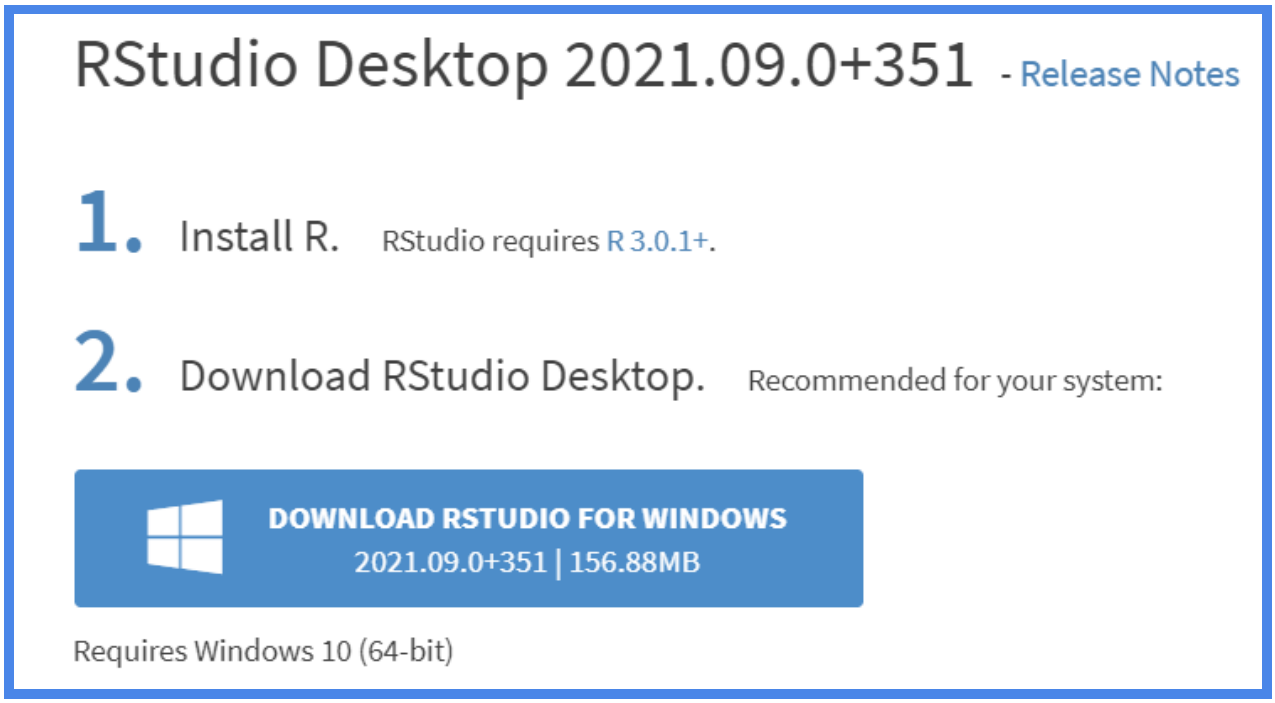
Note that Rtools is only needed build R packages with C/C++/Fortran code from source. By default, R for Windows installs the precompiled "binary packages" from CRAN, for which you do not need Rtools.

To use rtools, download the installer from CRAN:

- On Windows 64-bit: rtools40v2-x86_64.exe (includes both i386 and x64 compilers)
- On Windows 32-bit: rtools40-i686.exe (i386 compilers only)

Once the download is complete double-click to launch the installer. Choose "Next" in the command prompts to install.

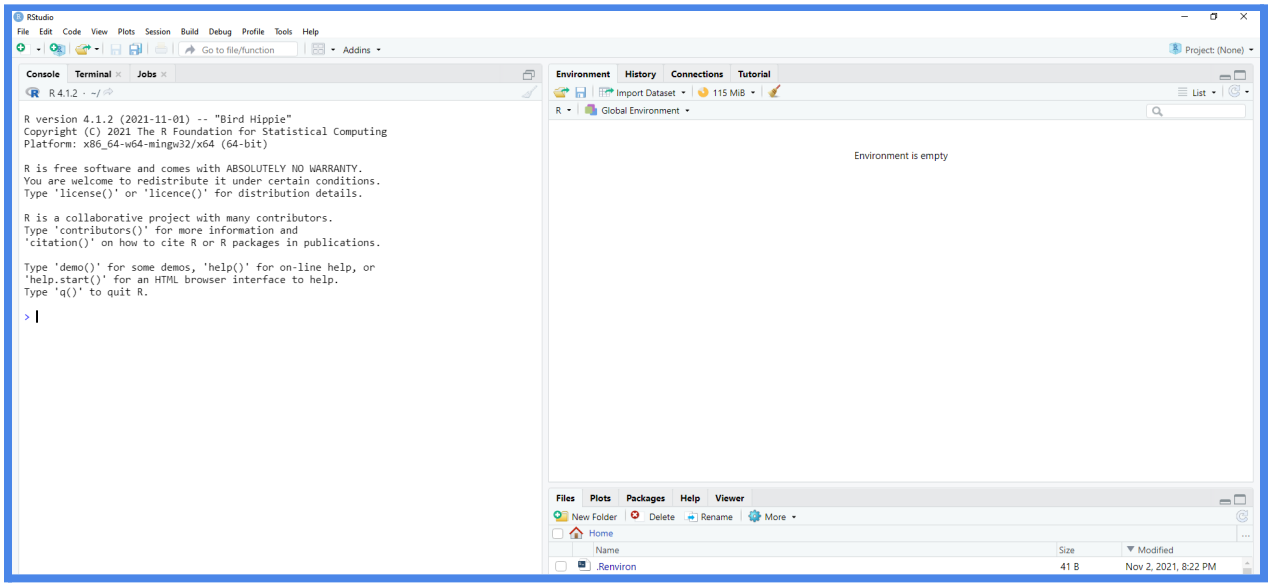
Now we will install RStudio. (https://www.rstudio.com/products/rstudio/download/#download)



Choose Download RStudio for Windows, and once the file is downloaded open the installer. Click Next through the installer to finish the installation.

Finally, we will confirm that everything has been installed correctly. We will be entering some code into R, which I won't explain here but it will become more familiar during the workshop.

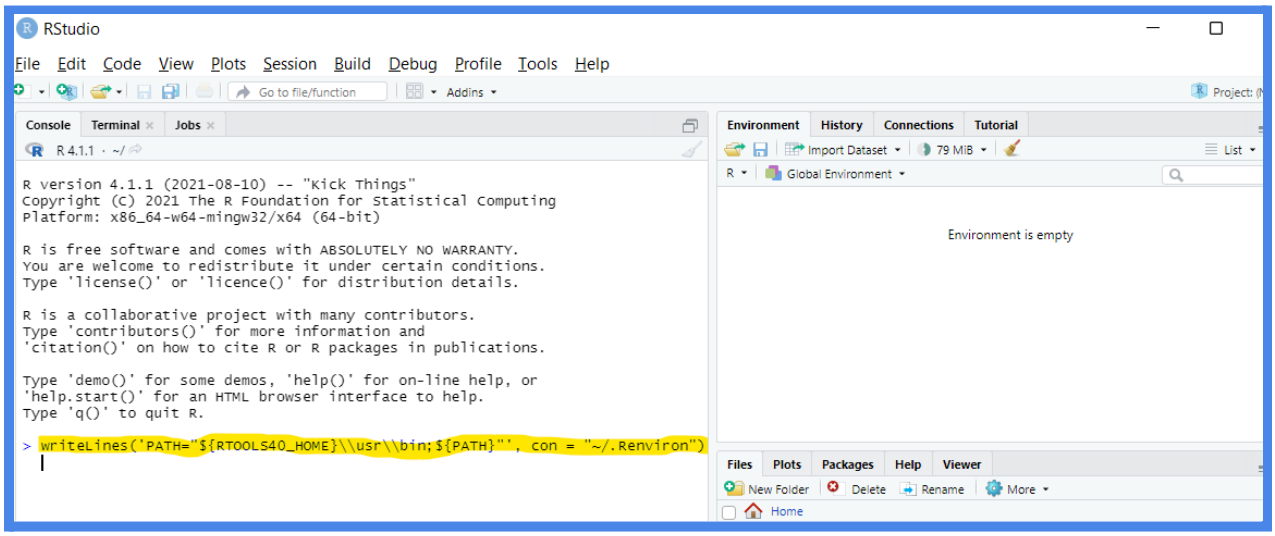
Open RStudio. You should see a window that looks similar to this:



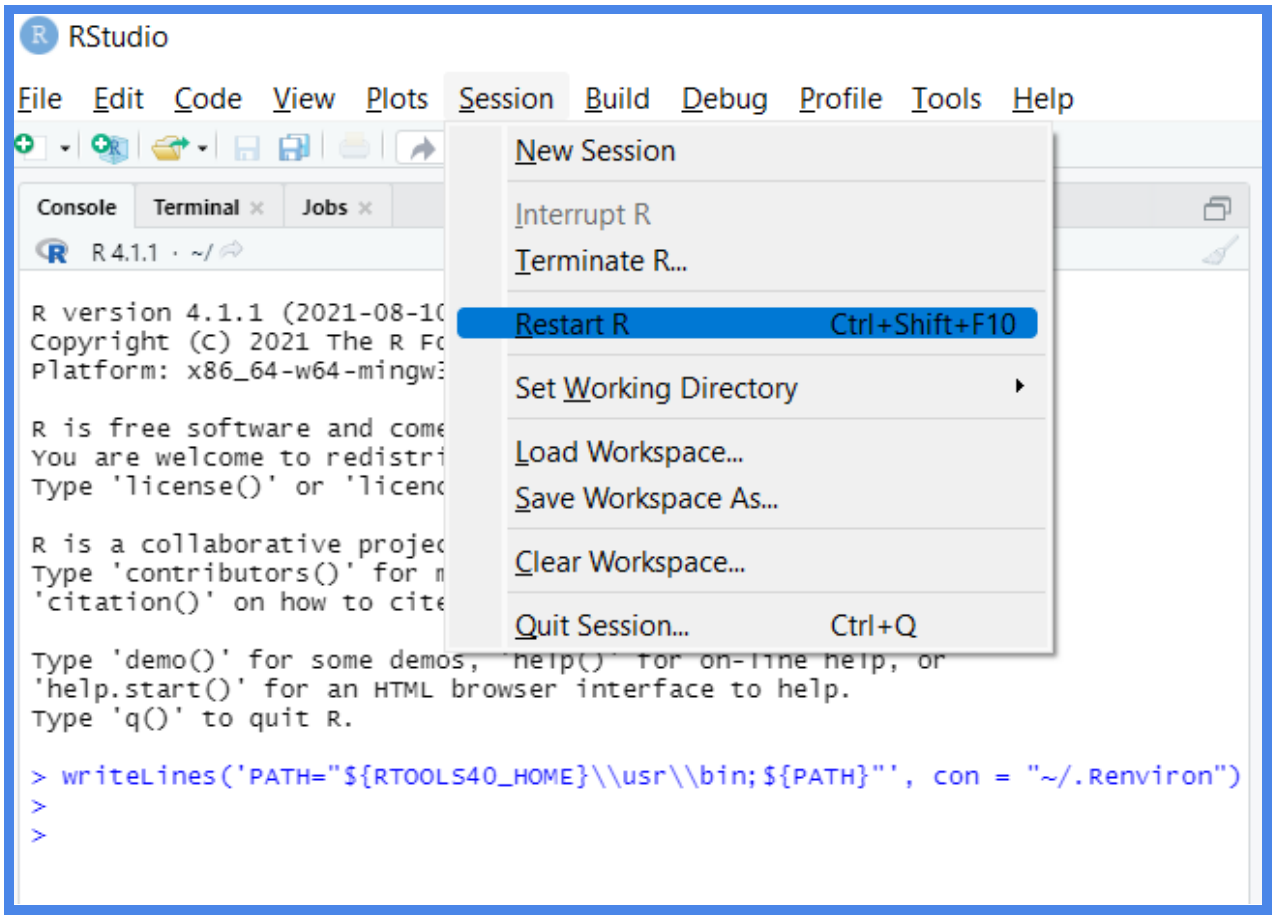
First we need to connect R and R Tools by writing the following line in the Console:

```
writeLines('PATH="${RTOOLS40_HOME}\\usr\\bin;${PATH}"', con = "~/.Renviron")
```

Then press Enter to execute the command.



Now we will restart the R session using the Sessions drop-down menu



Now we will run some checks to confirm that everything is properly set up. First run `Sys.which("make")`. This should return the path to an R Tools library:

```
Restarting R session...

> Sys.which("make")
                                    make
"C:\\rtools40\\usr\\bin\\make.exe"
>
```

Next, we will install an R package that depends on R Tools. Run `install.packages("jsonlite", type = "source")` in the Console. This should install the library and end with something similar to this:

```
* DONE (jsonlite)

The downloaded source packages are in
        'C:\Users\Justi\AppData\Local\Temp\RtmpmyiTZB\downloaded_packages'
>
```

Finally, let's install some packages that we will be using during the workshop. Run the following line in the console:

```
install.packages(c("tidyverse", "lubridate", "sf"))
```

This may take some time depending on your internet connection. Press OK for any prompts that appear. At the end you should get an output that looks similar to this:

```
trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.1/tidyverse_1.3.1.zip
Content type 'application/zip' length 430184 bytes (420 KB)
downloaded 420 KB

trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.1/lubridate_1.8.0.zip
Content type 'application/zip' length 1716146 bytes (1.6 MB)
downloaded 1.6 MB

trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.1/sf_1.0-3.zip'
Content type 'application/zip' length 42002019 bytes (40.1 MB)
downloaded 40.1 MB

package 'tidyverse' successfully unpacked and MD5 sums checked
package 'lubridate' successfully unpacked and MD5 sums checked
package 'sf' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\jmillar\AppData\Local\Temp\RtmpWIjqIq\downloaded_packages
```

Finally, check to make sure that the library has correctly installed by running `library(tidyverse)` in the Console. You should see and output similar to this:

```
> library(tidyverse)
-- Attaching packages --------------------------------------- tidyverse 1.3.1 --
v ggplot2 3.3.5     v purrr   0.3.4
v tibble  3.1.5     v dplyr   1.0.7
v tidyr   1.1.4     v stringr 1.4.0
v readr   2.0.2     v forcats 0.5.1
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
```

If all of this worked then everything should be ready to go! If you have any issues for questions please reach out to Justin Millar (jmillar@path.org) or Hannah Slater (hslater@path.org).