**Introduction to R - Program editors**

Often, you will have a long list of commands that you would like to execute all at once – i.e., a program. Instead of typing all of the code line by line at the R Console prompt, you could type it in Notepad or some other text editor and copy and paste the code into R.

R’s program editor

R has a very limited program editor built into it. Select FILE > NEW SCRIPT to create a new program. Below is what the editor looks like with some of the past examples.



To run the current line of code (where the cursor is positioned) or some highlighted code, select EDIT > RUN LINE OR SELECTION.

To run all of the program, select EDIT > RUN ALL. To save your code as a program outside of R, select FILE > SAVE and make sure to use a .R extension on the file name. To open a program, select FILE > OPEN SCRIPT. Note that you can have more than one program open at the same time.

There are MUCH BETTER program editors! The editors described next have color coding of the program code which makes reading programs MUCH easier! **I recommend using one of these editors.**

RStudio

RStudio Desktop (<http://www.rstudio.com>) is by far the most widely used editor for R. Its market penetration is so dominant that new R users often think of R and RStudio Desktop as the same. Instead, RStudio (http://www.rstudio.com) is a for-profit company. Due to its dominance, there has been some backlash toward its business practices.

After recording the video: The RStudio company has changed its name to Posit.

For the remainder of these notes, we will refer to RStudio Desktop as just “RStudio” due to how it is most commonly referred to as. RStudio runs on all major operating systems.

Below is a screen capture of RStudio where a program is already opened in it.



You can start a new program by selecting FILE > NEW FILE > R SCRIPT or open an existing program by selecting FILE > OPEN FILE (without a program open, you will not see the program editor). To run a segment of code, you can highlight it and then select the “Run” icon in the program editor window.

Also, the editor can suggest function or package names from any loaded package if <Tab> is pressed at the end of any text string. For example, typing “pn” and pressing <Tab> yields a pop-up window suggesting functions pnbinom(), png(), and pnorm(). Pressing <Tab> where an argument could be given within a function (e.g., after a function name and its opening parenthesis or after a comma following another argument) gives a list of possible arguments for that function.

The windows available on the right side of the screen provide some additional useful information. In the upper right corner, you can view the list of objects in R’s workspace (similar to using ls() or objects() in the R Console). In the bottom right corner, all graphs will be sent to the PLOTS tab and help is immediately available through the HELP tab. Also, in the bottom right corner window, packages can be installed via the PACKAGES tab.

Tinn-R

Before RStudio was created, there were a number of program editors available without necessarily one having a dominant position among users. One of these editors that continues to be developed is Tinn-R (<https://nbcgib.uesc.br/tinnr/en>). This is a free, Windows-based program editor that is a separate software package outside of R.

Below is a screen capture of Tinn-R where a program is already opened in it and important items are numbered.



To run code from the editor, R's GUI needs to be open. This can be opened by selecting the R control: gui/PuTTY (start/close) icon from the R toolbar (see #1 in the screen capture). Tinn-R subsequently opens R in its SDI (single-document interface), which is a little different from R's MDI (multiple-document interface) that we saw at the beginning of the R introduction notes.

Program code in Tinn-R can be transferred to R by selecting specific icons on Tinn-R's R toolbar. For example, a highlighted portion of code can be transferred to and then run in R by selecting the R send: selection (echo = TRUE) icon (#2). Note that the transfer of code from Tinn-R to R does not work in the MDI.

Below are some additional important comments and tips for using Tinn-R:

* Upon Tinn-R's first use with R's SDI, the TinnRcom package is automatically installed within R to allow for the communication between the two softwares. This package is subsequently always loaded for later uses.
* When R code is sent from Tinn-R to R, the default behavior is for Tinn-R to return as the window of focus (i.e., the window location of the cursor) after R completes running the code. If Tinn-R and R are sharing the same location on a monitor, this prevents the user from immediately seeing the results in R due to it being hidden behind the Tinn-R window. To change this behavior, select OPTIONS > APPLICATION > R > Rgui/PuTTY and uncheck the Return to Tinn-R box. Alternatively, select the “Options: return focus after send/control Rgui” icon on the Misc toolbar (#3).
* By default, the line containing the cursor is highlighted in yellow. To turn this option off, select OPTIONS > HIGHLIGHTERS (SETTINGS) and uncheck the ACTIVE LINE (CHOICE) box.
* Long lines of code are wrapped to a new line by default. This behavior can be changed by selecting OPTIONS > APPLICATION > EDITOR > DISPLAY and then selecting the NO radio button for LINE WRAPPING.
* Syntax highlighting can be maintained with code that is copied and pasted into a word processing program. After highlighting the desired code to copy, select EDIT > COPY FORMATTED (TO EXPORT) > RTF. The subsequently pasted code will retain its color.
* The style of syntax highlighting can be changed by selecting OPTIONS > HIGHLIGHTERS (SETTINGS). I have found it to be especially helpful to use background colors with particular code types, like comments, to make code quite distinct.
* Tinn-R can run R within its interface by using a link to a terminal version of R rather than R's GUI. I prefer not to use R in this manner.

When using Tinn-R and R's GUI, I have found it to be more efficient to view each window side-by-side in a wide-screen monitor or in a multiple monitor environment. This allows for both to be viewable at the same time. Code and output can be side-by-side in large windows without needing to switch back-and-forth between overlaying windows.

Other editors

Jupyter Notebook is part of Project Jupyter (<https://jupyter.org>) that provides free, open-source software for individuals to write/run computer programs. The name “Jupyter” is derived from the three languages originally used with it–Julia, Python, and R. While Jupyter Notebook can be downloaded and installed by users on their own computer, the operating system must be Linux. Windows users can still use Jupyter Notebook via a central Linux-based server through JupyterHub. Many universities provide access to students/faculty in this manner.

I have often used WinEdt with the RWinEdt add-on in the past on my Windows-based computers. Also, the Emacs editor (<http://www.gnu.org/software/emacs>) with the Emacs Speaks Statistics ([http://ess.r-project.org](http://ess.r-project.org/)) add-on are popular for Linux users.