Editing R and Stata files using a light-on-dark display in Notepad++

Sean Higgins[1]
http://www.seankhiggins.com

July 12, 2016

A light-on-dark display refers to the use of white and colored text on a black background. Below are instructions to edit your `.ado`, `.do`, and `.R` files using a light-on-dark color scheme in the feature-rich text editor Notepad++ and send these files directly from Notepad++ to R and Stata.

1. Install Notepad++ (http://notepad-plus-plus.org/).
2. In Notepad++, go to the Settings tab and select Style Configurator.
3. Under "Select theme" choose Twilight.
4. For some reason, the default background for text in the Twilight is dark gray even though the overall background is black, and if you highlight text in your file the highlighting is also gray so it is hard to see. To fix these issues, make sure Global Styles is selected under "Language" (still in the Style Configurator dialog box) and do the following:
   a) Under "Style" select Default Style, and under "Colour Style" make sure Background colour is set to black. Even if the colored box next to Background colour looks black, it might be dark gray so manually change it to black.
   b) Under "Style" select Selected text coulour and change background color to a light blue. I prefer RGB 0:128:255 which is in the first row (third from the right) of the color palette that appears if you click on the colored box next to Background colour.
5. Configure syntax highlighting. For programs that are not included in Notepad++'s Language menu, the syntax coloring has to be told to Notepad++ manually through an `.xml` file. I created an `.xml` file specifically for use with the Twilight light-on-dark color scheme. Download `userDefineLang.xml`[2] then install this file in the right location:
   a) Press the windows key + R to open the Run dialog box.
   b) Under "Open" enter %APPDATA%\Notepad++
   c) Click OK or press enter.

[1] If you are unsuccessful in configuring R or Stata with Notepad++ and a light-on-dark display using these instructions, please email me at seanhiggins@berkeley.edu so I can look for a solution and fix these instructions.

[2] For Stata, it is adapted from the normal (dark text on white background) syntax highlighting for Stata available at http://notepad-plus.sourceforge.net/commun/userDefinedLang/userDefineLang_stata.xml.

d) Copy the `userDefineLang.xml` file into the folder that opens up. Do not change the name of the file! For some reason Notepad++ won't read the file if you change the name.

6. Open Notepad++, or if you already have it open, close and reopen the program. Open a script and look at the pretty colors. R code should look like this:[3]

```
    # READ IN SHAPE FILE
    myshp = readShapePoly(paste0(state,"/",state,suffix),
        proj4string = CRS("+proj=longlat +datum=WGS84"))

    # GENERATE CENTROIDS
    centroids = gCentroid(myshp,byid=T)
    class(centroids)

    # PLOT TO TEST THAT IT WORKED:
    if (plot==TRUE) { # runs slower but makes plots so you can see how the
                  # through localities
        plot(myshp)
        plot(centroids,add=T,col="red")
    }

    # MAKE A SPATIAL POINTS DATA FRAME WITH THE BLOCK CODES AND CENTROIDS
    centroids.spdf = SpatialPointsDataFrame(centroids,data=myshp@data)
```

Stata code should look like this:

```
program define randomselect, sortpreserve
    syntax [if] [in], n(real) gen(string) [select(string) seed(real -1)]

    // Preliminaries
    tempvar random tag pregen touse
    gen `touse' = 0
    replace `touse' = -1 `if' `in'

    // Seed
    if `seed'!=-1 set seed `seed'

    // Random
    gen `random' = runiform()
    if "`select'"=="" {
        sort `touse' `random', stable // so touse==1 goes on top
        gen byte `gen' = (_n<=`n')
    }
    else {
        sort `select', stable
        by `select' : gen `tag' = -(_n==1) if `touse' // mark one obs per
        sort `tag' `random', stable // -`tag' to get the =1 at the top
        gen byte `pregen' = (_n<=`n') if `touse' // `pregen' will = 1 for
        sort `select', stable
        by `select' : egen `gen' = max(`pregen') if `touse' // now tag all
    }
```

7. Set up a way to send your `.R` files directly to R.

a) Follow these instructions for using NpptoR for this purpose.

[3] If the syntax highlighting doesn't match the picture, it might be because a different syntax highlighting for R comes with Notepad++. Go to Settings > Preferences > Language Menu, find R in the list, and click the ⇒ button to move it to the "Disabled items" list so that the built-in syntax highlighting is disabled and the `.xml` file is read instead.

b) After following his instructions, you should be able to run an entire `.R` file with Ctrl+F8, or part by highlighting the lines that you want to run and pressing F8.

8. Set up a way to send your `.do` files directly to Stata.

   a) Follow Friedrich Huebler's instructions on http://huebler.blogspot.ca/2008/04/stata.html

   b) After following his instructions, you should be able to run an entire `.do` file in Stata by pressing F10, or part of a `.do` file by highlighting the lines that you want to run and pressing F9.

You are ready to go! Notepad++ can also be set up for many different tasks with the same color scheme (I use it for Python, LaTeX, and batch files). An advantage of using Notepad++ for multiple programming languages is that the syntax highlighting will always look similar, e.g., comments will be green in any script. The shortcuts are also the same, e.g., Ctrl+K turns a line into a comment, and Notepad++ automatically knows what character to add to do this (% in LaTeX, ** in Stata,[4] # in Python and R, etc.). Contact me at seanhiggins@berkeley.edu if you have questions.

[4]Notepad++ can't differentiate between when * is used to begin a comment and when it is used as a multiplication sign or wildcard. As a result, I have changed the comment code that Notepad++ recognizes to ** in the attached `userDefineLang.xml` file (props to Bibek Adhikari for the suggestion). This has the advantage that both Stata's do-file editor and Notepad++ will recognize lines beginning with ** as comments. If we instead specified * to denote a comment in `userDefineLang.xml`, Notepad++ would color everything after a wildcard or multiplication * as a comment. The only drawback to this solution is that if you open other people's do files that use * for comments, those lines will not be colored green.