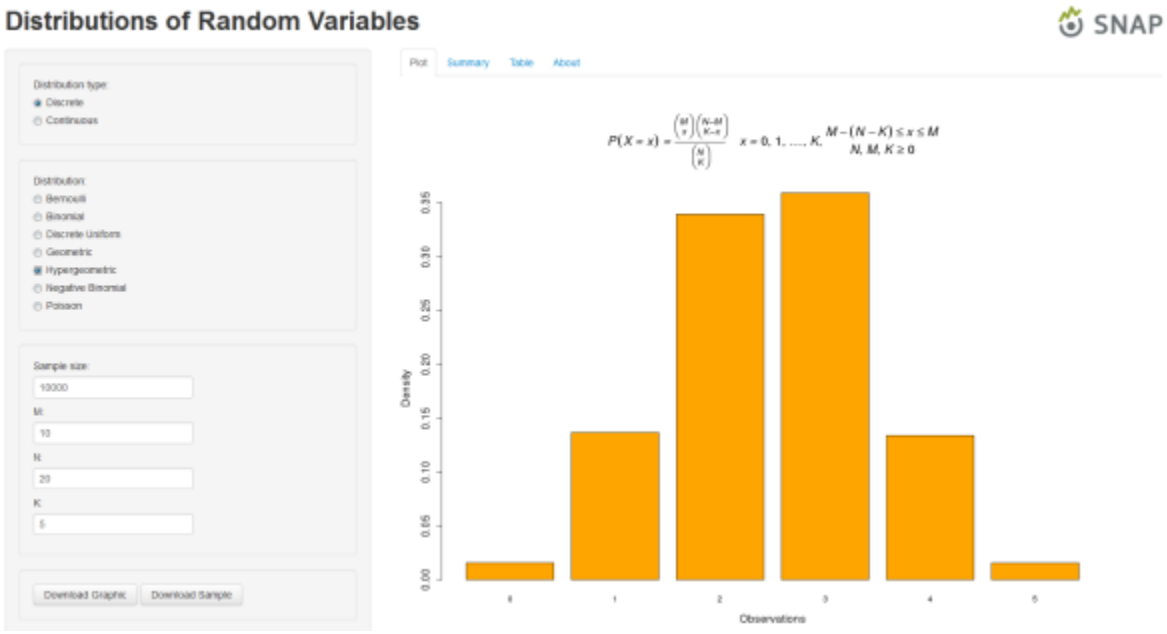
R sampling app version 4

Posted on **May 20, 2013** by **Matt Leonawicz**

Continuing from my previous post, R sampling app version 3 [app], here I expand upon the example Shiny app that was presented, which generates random samples from some common probability distributions. I have made a number of changes, big and small, from the previous version.



You can click the image to go to the app page.

The changes: First, I added radio buttons to the sidebar to allow the user to toggle between discrete and continuous distributions. This cleaned up the sidebar panel a bit by only displaying the distribution names for one group of distributions at a time. I also added `if(length(input$dist))` statements to `output$dist1`, `output$dist2`, `output$dist3`, and `output$plot` to reduce errors. The errors did not break the app, but it wasn't professional to allow those brief flashes of red text to be printed to the screen when first launching the app, which simply resulted from the fact that there is a moment now at launch where input$dist is not yet set to a distribution name.

Next, I elected to retain the option of sample density curve overlays for continuous distributions only. I also increased the default sample size and switched from `sliderInput` to `numericInput` in the `ui.R` script. I added `wellPanel` calls to the `ui.R` script to organize/segment the sidebar panel a bit. The final changes included adding a download button for obtaining a pdf of the currently displayed plot and an 'About' tab that gives more information about the app. The 'About' tab is sourced in from an `about.r` file.

See my posts, Mathematical notation in R plots and Mathematical notation in R plots 2, for some examples and conveniently supplied **R** code for displaying the equation of a pdf or pmf on an **R** plot for many common distributions. The code shown in this post for version 4 of the random sampling app will make use of all these **R** expressions, but you won't see the expressions explicitly, so please see the other posts if you are interested in how they work or just copying them for your convenience.

Here is the code that is run to build the workspace file that will be loaded by the app:

I save this workspace as `samplingApp.RData` and place it in the directory that contains the `ui.R` and `server.R` scripts. Next, we have the updated `ui.R` script.

Next we have the updated `server.R` script.

Finally, the new script addition is the `about.r` file. I've saved it as a function that can be sourced in and passed to `tabsetPanel`. I know this would probably be considered a very odd way to do this. Some might say, "just make this part in straight up html and source it with the shiny package `HTML` function." Well, the mood struck me to just make it a third **R** script and not add any other file types, even if it is all just text. And as is invariably the case with me, "I already had **R** running." Heck, sometimes I even use **R** to send system calls to VLC media player when I'm listening to music. I mean, that darn **R** console is always right in my face; there's no getting around it!