Using python in R markdown

Installing `reticulate` to run python

Need to install the `reticulate` package to run python code in Rstudio.

```
# install.packages("reticulate")
```

Installing python (if you don't have it installed already)

If you have python installed already, skip this. The easiest way to install python for use in Rstudio (if you do not already have python installed) is:

```
reticulate::install_miniconda()
```

In R studio, you can press CTRL+ENTER (CMD+ENTER) when your cursor is on the line above to execute the line. Even though the chunk is set to `eval=FALSE` in the .Rmd file, you can still execute it line-by-line with Ctrl+ENTER (or CTRL+SHIFT+ENTER).

Using R packages

There are two ways to use R packages. After you've installed a package, you can refer to any function or value inside that package using the `::` notation. In the chunk above, `reticulate::install_miniconda()` is saying "run the `install_miniconda()` function from the `reticulate` package.

You can also import all the functions from a package at the same time using `library(packagename)`. This allows you to just use the name of the function. So we could have instead used `library(reticulate)` then `install_miniconda()` in the chunk above. There are generally two concerns when deciding when to use one method or the other:

- How many times will you be using functions from this package. If it's a lot, it's easier to run `library(reticulate)`.
- Does function from that package conflict with other function names? For example, if `obscure_package` is a package with a plotting function you want to use, you could use `library(obscure_package)` then `plot()` from that package. `plot()` will now cover up the usual `plot()` function. But using `obscure_package::plot()` will not cover up the default and you can still use the `plot()`. Also, technically every function lives in a package and you can see the package it belongs to using the `?function_name` command or help window. You can always use the default functions by referring to their package (e.g., the default plot function lives in `graphics` and can be accessed using `graphics::plot()`).

It's generally hard to know the names of all functions in a package though. Luckily, when you run `library(obscure_package)`, R tells you what functions this package is covering up. Try `library(tidyverse)` to see the functions that the `tidyverse` package covers up.

Selecting a python interpeter (optional)

We can to select a python interpreter (the program that executes the python code). If no selection is made, then the default python interpreter will be used (which is generally the most updated version of python installed). This is the python that activates when you type `python` into a terminal. If you have different conda environments, then you can select the python interpreter for a specific environment using the steps below. (sidenote: it's suggested that you different conda environments for different projects so you don't accidentally install any python packages that mess up the way other programs run. You can either keep all your conda environments in one place, or store the environment in the project folder – Rstudio will use this version of python by default.)

To select a python interpreter: - open the Glboal Options (Tools > Global Options) - click the `python` section, click `Select...` - Click the `Conda Environments` tab to select the conda environment you wish to use.

Running python in Rstudio

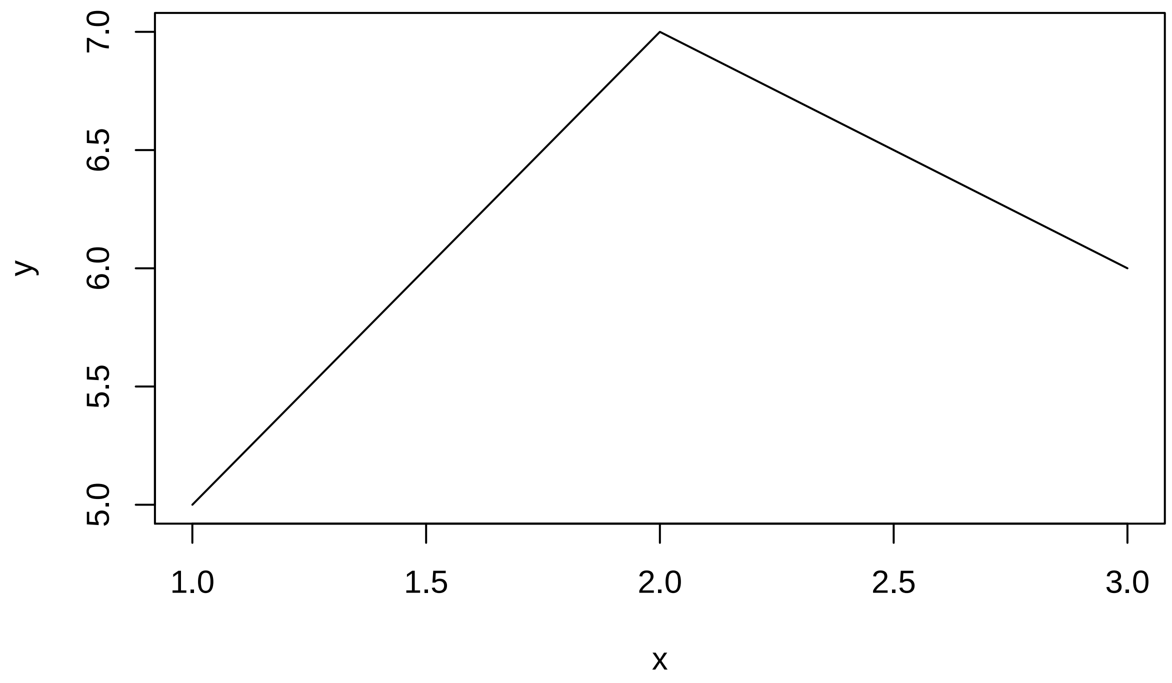
An R code chunk in an R markdown (.Rmd) file looks like this (backticks, not apostraphes):

```
‘‘‘{r [Name of Chunk], [compile options]}
code here
‘‘‘
```

Note the little `r` right after the first bracket – this controls which language we use to interpret this chunk. Below is a valid R chunk (if `x` and `y` are already defined):

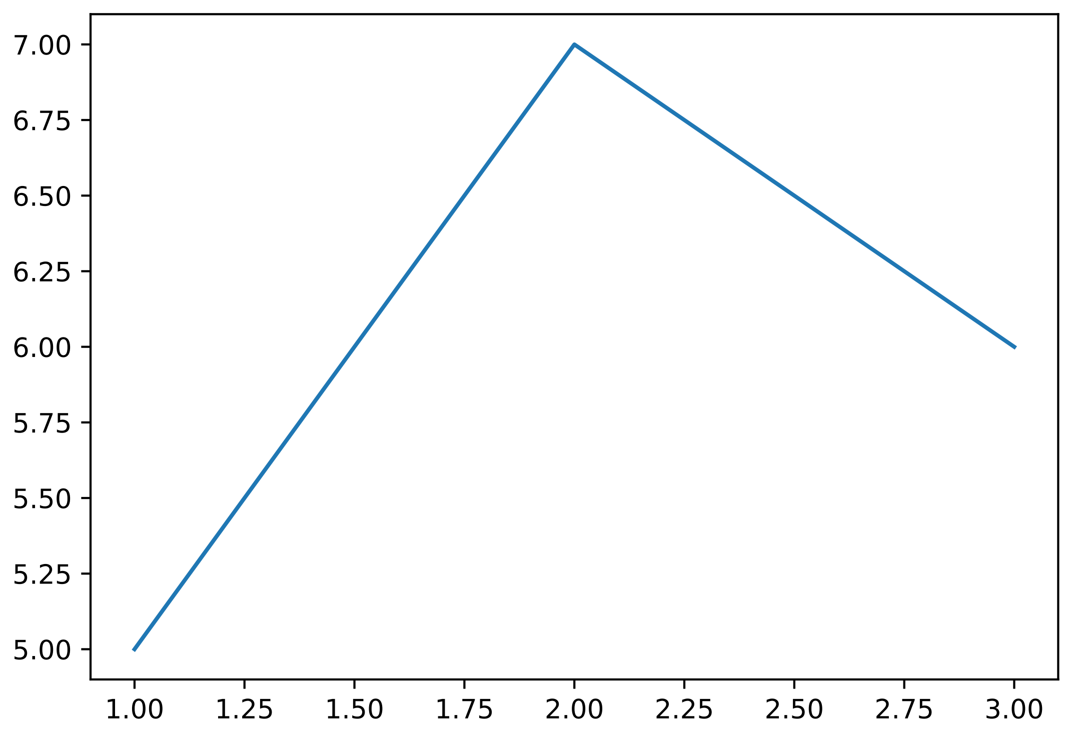
```
‘‘‘{r Plotting my data, eval=F}
x = c(1,2,3)
y = c(5, 6, 7)
graphics::plot(x,y, 'l')
‘‘‘
```

Note that `graphics::` is uneccesary here because it's loaded by default, I just included it to compare to using python packages below. The `eval=F` tells R not to evaluate/compile this chunk. If you leave that option off, the default is to evaluate the chunk. To run a python chunk, we want to put `python` right after the brackets.



~~~
```{python Plotting my data, eval=TRUE}
import matplotlib.pyplot as plt
x = [1,2,3]
y = [5, 6, 7]
plt.plot(x, y, '-')
```
~~~

Note that there is no default plotting package in python, so we need to use the matplotlib package.



R Markdown

Some great features of R Markdown:

- Table of contents on the right of the code pane to select by header
- Code chunk selection below the code pane
- line-by-line or full-chunk execution
- running different languages in the same file
- can export R markdown files to .R and .py scripts if needed
- can “knit” to HTML, PDF, MS Word
- can use `browser()` to pause midway through execution

Some great features of Rstudio:

- Git / GitHub integration
- help window with documentation on packages
- useful keyboard shortcuts
- debugging (better in R scripts than R markdown)