Notes on using Python within RStudio

Christopher Paciorek

February 2022

In addition to basic use of Python chunks in R Markdown documents, RStudio has developed the `reticulate` package, which greatly enhances the ability to use both R and Python in a single workflow.

Rmd documents with Python chunks can be rendered to an output file by 'knitting' the document using the `Knit` button (which invokes `rmarkdown::render`) or by invoking `rmarkdown::render` yourself.

With the notebook (inline output) functionality turned off

If you run a Python chunk from an R Markdown document in RStudio it will run the code in a Python process and show the result in the Console window.

In fact it will invoke `reticulate::repl_python` to give you a "read-eval-print loop" (REPL) interpreter interface to Python in the console. If you want, you can enter Python code directly in the console. In addition, execution of subsequent Python code chunks can use objects from earlier chunks. To exit from the Python REPL to return to the R REPL, you can either type exit at the Python prompt or simply execute an R chunk.

With the notebook (inline output) functionality turned on

In the RStudio Global Tools options, under `R Markdown`, you can select `Show output inline for all R Markdown documents`. This causes the output from Python chunks (as well as chunks in other languages) to show up in the editor window (i.e., notebook style) rather than in the console.

Reticulate and working in both R and Python

`reticulate` allows you to work on objects in both R and Python, moving seamlessly between the two languages. Lots more detail in the reticulate documentation and online.

```
library(reticulate)
devs <- rnorm(5)
```

```
## access the R object in Python code
r.devs[0]
```

```
## 0.5846187696119165
```

```
pyvals = {"a": 0, "b": 1}
```

```
## access the Python object in R code
py$pyvals['a']
```

```
## $a
## [1] 0
```

There's lots more functionality. That's just a teaser of the cool things you can do.