**5 Quick and Easy Tips to Become an Efficient Data Scientist in Both Python and R**

Having difficulty transitioning between using Python and R? Try these tips to make the transition smoother



Photo by [Ivan Shunyakov](https://unsplash.com/@lamoshqa?utm_source=medium&utm_medium=referral) on [Unsplash](https://unsplash.com/?utm_source=medium&utm_medium=referral)

A recent discovery of mine is that many data scientists out there use both Python and R.

The story typically goes that they use R for graduate studies and research while using Python for work. Since many individuals work while going to school, these data scientists have to switch back and forth between Python and R, sometimes several times within a single day depending on what they’re working on for school or work.

Because many (read: most) data science courses or learning paths focus on one language over another, there’s a gap in the learning that neglects to help data scientists switch between both languages. Perhaps surprisingly, switching between two programming languages on the same day can be challenging, especially when you’re seeking to complete similar work. For example, it’s easy for web developers to switch between HTML, CSS, and JavaScript in the same day because each language handles a very specific piece of a website. However, data scientists can be completing one type of data analysis in R for a graduate studies class while later in the day they need to use Python to complete a different analysis for their job.

The high number of data scientists who are switching between both languages in a given day means that a gap in data science education needs to be filled by providing tips on how to better manage using both languages regularly. Here, we’ll look at a few simple tips that can be implemented as part of your routine in less than 10 minutes that will help you become a more efficient data scientist at switching between Python and R.

**1. Complete your Python and R tasks on separate days**

One of the best things you can do to begin working efficiently in both Python and R is to complete your Python and R work on separate days. By batching all of your Python tasks on one day and R tasks on another, you guarantee that you won’t need to switch languages in the same day.

This takes less than 10 minutes to implement as all you need to do is plan your week around what language you will be working with on a given day. For example, Monday, Wednesday, and Friday could be days you complete work done in Python, whereas Tuesday and Thursday are reserved for tasks requiring R.

Day-batching your tasks based on programming language allows you to remain in a state of flow where you don’t have to worry about using different conventions, syntax, or documentation. Instead, you can focus on writing clean code given the language’s conventions without falling into the trap of coding with an accent (see tip 5).

I remember in my first year of university studying software development that most of my coding courses used C#. This made it easy to move from class to class without having to get in the groove of using a new language. Things got substantially harder in my second year when we began taking courses in a variety of languages, including C#, Java, PHP, and more. This made it more difficult to get into a groove due to constantly having to change programming languages every two hours.

This challenge is comparable to speaking several different verbal languages in a day. Before you become fluent, it’s mentally taxing to make the switch between different languages. The shapes your mouth and tongue need to make are different, the sounds are different, the grammar is different, and the conventions and customs of the language are different. However, if one day you speak English, the next French, the next Italian, and so on, you’ll find that you’re much more capable of speaking efficiently and effectively because you’re working for an extended period of time in one language.

However, it’s not always possible to dedicate entire days to either Python or R. Sometimes, work, school, and life all happen at once which means that you need to juggle several different deadlines, projects, and requirements. An alternative, though similar method, of batching your tasks is to make one-half of the day dedicated to Python tasks and the other half dedicated to R tasks. Then, by having a substantial break in the middle where you do nothing related to either batch of tasks, you can move into the latter half of the day ready to work in a different language. The break in the middle of the day is essential to reset your brain and give it a chance to expunge your usage of the other language. By blocking your tasks into two separate halves of the day, you ensure that you’re not working on one task in Python for 20 minutes, followed by a task in R for one hour, and a continuation of this cycle that leaves your brain exhausted and not working at its maximum efficiency.

**2. Find the similarities between Python and R**

When I was in university, I had the easiest time working in both C# and Java because I was able to find the similarities between the two languages and use them to work efficiently in both languages. Similarly, when I was learning Russian, I found several similarities between it and French and English, both of which I already knew. These similarities allowed me to understand the connotation of a lot of words even if I didn’t understand the words immediately at first.

The trick to being able to switch between Python and R is to find the similarities between the two languages, however finite they might be. For example, both Python and R can be used to write object-oriented code and they both can be used to conduct statistical analyses. Sure, working from two differently-indexed languages can be a pain, but at least they have more in common than is different. However, you can at least use the same IDE for both languages, as RStudio and Atom are capable of supporting both R and Python.

It’s important to remember that the similarities you find between the two languages may be different from those found by someone else. Similar to how analogies can be created in different ways to describe the relationship between two things, so too can the similarities you find between R and Python be used to ease the transition between both languages.

This tip takes less than 10 minutes to complete as you can begin creating a list of Python and R similarities on a sticky note. This note can be left on your desk as a reminder of the key points of each language that can be used to easily transition from one to the other.

**3. Learn how to ask/Google/StackOverflow the right questions**

The most valuable skill I learned while studying software development at university is how to ask good questions. I’m sure you can relate that asking ill-defined questions will lead you to about 200 different StackOverflow hits that may or may not sort of relate to the answer you’re looking for but are not exact enough that you get a quick solution to your issue.

Working in two languages like Python and R requires that you can ask the right question for the language you’re currently working in. For example, asking a question about R in the same way you would ask a question about Python may return results that, as mentioned previously, aren’t quite what you were looking for.

While copying error messages into search bars is a great way to find StackOverflow responses to similar problems. However, do you really understand what went wrong if all you’re doing is copying an error message? A better way to become familiar with the quirks of Python and R is to work towards understanding what went wrong in your code. Then, you can ask better questions that apply more directly to the problem you’re having than a generic error message ever could.

Practicing how you ask your questions can be completed in less than 10 minutes each day and is best completed while you’re working. You’ll quickly begin to notice which questions are yielding the right answers versus those that are leading you to general answers that could solve any myriad of problems.

**4. Keep technical documentation handy and learn how to use it effectively**

While I mentioned earlier that Python and R have more in common than not, one of the key places where they differ significantly is their documentation.

Python is known for having some of the best code documentation out there, whereas R is seriously lacking. Keeping whatever documentation you can get your hands on close by can help you write code efficiently when transitioning between two languages. For example, during university when I was working on a project in Typescript (Angular, to be exact) I kept its documentation open in a browser, ready for whenever I hit a gap in my knowledge. While Typescript languages are similar to JavaScript, I found that there was enough of a learning curve between the two languages that having the documentation handy at all times made the transition that much smoother.

However, I’ve always found that creating my own documentation or cheat sheets is a million times more valuable than looking up technical documentation, no matter how well-written. You know how you end up subconsciously studying while creating cheat sheets or study guides, even if it doesn't feel like it? The same thing applies when creating reference guides, documentation, sheet sheets, or study guides for Python and R. I’ve seen some beautiful reference sheets shared on LinkedIn that condense a vast amount of knowledge down into a single 8.5 x 11 sheet of paper that you can print out and have beside you at all times. These reference sheets can be used as inspiration and a starting point to begin creating your own cheat sheets and documentation that can help you make the transition between Python and R. I like including code snippets, formulas, function information, and package descriptions in my reference sheets for different programming languages.

Creating your own reference sheet or nabbing someone else’s can take less than 10 minutes and can save you the need of having to memorize the intricacies of both Python and R.

**5. Don’t code with an accent**

Speaking with an accent while working on a new natural language is cute. Coding with an accent is not.

I’ve [said this before](https://towardsdatascience.com/how-to-become-fluent-in-multiple-programming-languages-9f473c146b90) and I’ll say it again: it doesn’t do you any good to code with a language as if it were another language.

When you code in a language not using it the way it was intended, it will be unreadable by anyone else using it, it won’t run the way it was intended to run, or worse, it won’t even know what it is that you’re asking it to do.

Becoming deeply familiar with how Python and R are uniquely designed, their conventions, and their strengths and weaknesses will help you write code in each language to the best of its potential. This is where having quick reference sheets (see tip 4 above) can be handy as they provide you with the correct syntax, variable conventions, and format that's standard for the language you’re working in.

Python and R are pretty dissimilar when it comes to their syntax, format, and conventions, which is why having a reference guide can be handy to keep you from slipping into coding with an accent.

My favorite way to make sure that I’m not coding with an accent is to set an alarm for every half hour or hour of work that I complete that reminds me to check my code for fluency. This code check takes less than 10 minutes and is just a reminder for you to look over syntax, conventions, and format to ensure that you’re writing Python while working in Python or that you’re writing R while working in R.