title: Introduction to R and Python
subtitle: Command Line R and Python
minutes: 20

R and Python

> ## Learning Objectives
>
> - Understand the similarities, differences, strengths, and weaknesses of R and Python.
> - Interact through the shell with the R and Python interpreters.
> - Assign and understand variables.

Which should I learn?

While it is very important to stress that you can do nearly everything you do in one language in the other, each language certainly excels at different tasks. It is important to remember the philosophy behind each language, which is the reason for its development toward, and suitability for, different tasks.

Similarities:

- Interpreted
- High-level
- Object-oriented
- Large open-source repository (R: CRAN (https://cran.r-project.org), Python: PyPi (https://pypi.python.org/pypi))
- Widely used in data science and academia

R	Python
- Written *by* and *for* statisticians	- General purpose, easily written, easily read, easily learned
- R Studio (https://www.rstudio.com) GUI	- Text Editors, IDEs, and the Jupyter (http://jupyter.org) Notebook
- Quick statistical analyses	- Easy text (string) manipulation
- Beautiful, easy plotting (ggplot)	- More difficult plotting (matplotlib)
- More difficult webscraping and API work	- Easy, streamlined webscraping and API work, build websites and apps (Flask, Django)
- Though changing, R has traditionally been used in academia and data science	- Python is used in all sorts of programming domains
- Slower than Python at scale	- Faster than R at scale

See this blog post (https://www.datacamp.com/community/tutorials/r-or-python-for-data-analysis#gs.S_vlpQU) for a great comparison of the two languages!

R and Python in the shell

Let's open up two shells side-by-side, and type R in one and `ipython` in the other. Both can be used just like calculators, try some basic math such as `5 + 2` or `10 * 20` in either.

Defining variables in R and Python.

We can also store information by defining a *variable*, which we can then refer to whenever we want to use that value again.

In R this is done with <-:

```
> my_var <- 5
```

In Python, variables are assigned with =:

```
In [1]: my_var = 5
```

Now try the same mathematical operators using my_var and another number and see what happens.

We can also assign text to a variable. In R:

```
> my_name <- "Chris"
```

In Python:

```
In [2]: my_name = "Chris"
```

We can begin to see differences in terminology and thinking when we ask R and Python to tell us what type of data these variables hold:

In R:

```
> class(my_var)
```

```
[1] "numeric"
```

```
> class(my_name)
```

```
[1] "character"
```

In Python:

```
In [3]: type(my_var)
```

```
Out[3]: int
```

```
In [4]: type(my_name)
```

```
Out[4]: str
```

While both languages categorize the data, we can begin to see the languages diverge in how they understand data. This is only the beginning for R and Python, each have their own data types, which, though similar, have specific actions associated with them. This idea is the center of what is called *object-oriented programming*. The difference in these actions, and the additional libraries built upon them by the community, is what drives the philosophies and functionalities behind R and Python.