Python Intro — Part 2

Spectrograms

Daniel Guest

November 21, 2016

First, a small aside...

- ▶ This entire presentation is a KnitR demonstration, as well as a presentation on Python
- ▶ Check out the .Rnw file to see the mixture of LaTeX code and R code that is Sweaved together by KnitR to generate the output
- ▶ I'm using the LaTeX beamer package to create this slideshow
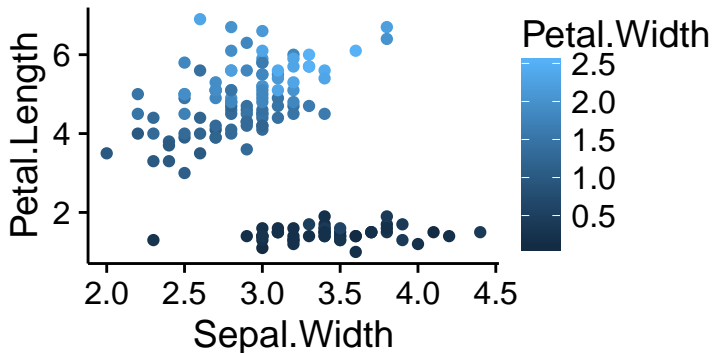- ▶ KnitR makes for very elegant presentations of R code and easy inclusion of graphics

First, a small aside...

```
d <- select(iris, Sepal.Width, Petal.Length, Petal.Width)
head(d)

##   Sepal.Width Petal.Length Petal.Width
## 1         3.5          1.4         0.2
## 2         3.0          1.4         0.2
## 3         3.2          1.3         0.2
## 4         3.1          1.5         0.2
## 5         3.6          1.4         0.2
## 6         3.9          1.7         0.4
```

First, a small aside...

```
ggplot(d,aes(x=Sepal.Width, y=Petal.Length, color=Petal.Wid
```

First, a small aside...

- Of course, the question arises — Why is KnitR and LaTeX in this presentation?
- KnitR is also ready to handle other languages
- If you ever need to present code in Python, C, R, Scala, Perl, Fortran, etc...
- For example...

```python
x = [1,2,3,4,5]
x.append(4)
print(2+2)
print(x)

## 4
## [1, 2, 3, 4, 5, 4]
```

Plan

- ► We're going to focus on two things today
  1. Theoretical — control statements and loops
  2. Practical — applying numpy and scipy to analyze sound
- ► What we'll need
  1. Python 3 (through IPython)
  2. numpy, scipy, matplotlib, sounddevice
  3. Sound files (in folder "samples")
- ► Overview
  1. Set up scripting tools
  2. A brief intro to loops
  3. Make one spectrogram
  4. Make a bunch of spectrograms

Bare bones scripting

- Today, we'll set up a bare bones environment for scripts
- You'll soon probably want something more fully fledged...
  1. Spyder
  2. Eclipse
- Now, let's open up Notepad++

Our first loop

- Loops in Python have a very easy and powerful syntax

```
for i in range(5):
 print(i)

## 0
## 1
## 2
## 3
## 4
```

Loop syntax

- Unlike MATLAB and R which employ brackets, Python uses tabs to indicate nesting
- If we don't nest at least one line for a loop we'll have problems
- I personally like this, I think it promotes more readable code!

```
for i in range(5):
print(i)

##   File "<string>", line 2
##     print(i)
##         ^
## IndentationError: expected an indented block
```

Looping over other things

- ▶ We can loop over a variety of objects
- ▶ The range() function is for when we want integer sequences, but many things in Python can be used to create a loop (technically, these things are called iterables in Python-lingo)
- ▶ Lists are iterables...

```
x = [1,4,9,16,25]
for i in x:
 print(str(i) + str(x))

## 1[1, 4, 9, 16, 25]
## 4[1, 4, 9, 16, 25]
## 9[1, 4, 9, 16, 25]
## 16[1, 4, 9, 16, 25]
## 25[1, 4, 9, 16, 25]
```

Looping over other things

- Strings are also iterables...

```
for i in "Hello":
 print(i)

## H
## e
## l
## l
## o
```

Setting up the environment

- First, we need to import the necessary modules

```
import numpy
from scipy.io import wavfile
import matplotlib.pyplot as plt
```

- Numpy — provides key matrix and DSP operations
- wavfile — provides read/write to wavfile
- pyplot — user-friendly functions and objects for plotting