Python Intro — Part 2

Loops, Spectrograms

Daniel Guest

August 13, 2017

KnitR

First, a small aside...

- This entire presentation is a KnitR demonstration, as well as a presentation on Python
- Check out the .Rnw file at `https://github.com/guestdaniel/python_intro/tree/master/02_spectrograms`
- KnitR mixes R and LaTeX to generate high quality publications and presentations
  - This makes it super easy to include graphics
  - KnitR also has nice options for syntax highlighting for many programming languages
- I'm using the LaTeX beamer package to create this slideshow

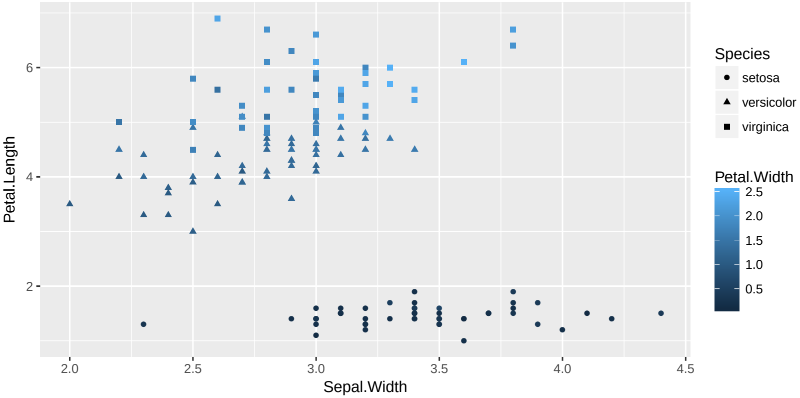
First, a small aside...

A nice demonstration of KnitR!

```
head(select(iris, Sepal.Width, Petal.Length, Petal.Width, Species))

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

head(iris %>% group_by(Species) %>% summarize(wid.mean = mean(Sepal.Width), wid.sd = sd(Sepal.Width)))

## # A tibble: 3 x 3
##      Species wid.mean    wid.sd
##       <fctr>    <dbl>     <dbl>
## 1     setosa    3.428 0.3790644
## 2 versicolor    2.770 0.3137983
## 3  virginica    2.974 0.3224966
```

First, a small aside...

```
ggplot(iris, aes(x=Sepal.Width, y=Petal.Length, color=Petal.Width, shape=Species)) + geom_point()
```



Intro

Plan

- We're going to focus on two things
  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. Get the packages we need
  2. Set up editor
  3. A brief intro to loops
  4. Make one spectrogram
  5. Make a bunch of spectrograms

Packages

Installing sounddevice

- Why we need each of the following packages is explained later!
- We should already have numpy, scipy, and matplotlib installed through Conda
- However, we do need to install sounddevice, which will allow us to play sound from a numpy array
- Unfortunately, Conda doesn't support installs of sounddevice, so we'll install it manually using pip
- pip is a popular Python package manager
- To do so, we'll run the following command (in your terminal, *not* in Python):

```
pip install sounddevice
```

Scripts

Editors

- Today, we'll just be writing scripts in a text editor, saving them to disk, and then running them through the IPython terminal
- You'll soon probably want something more fully fledged, these are good options:
  1. Spyder
  2. Eclipse
  3. SublimeText
  4. PyCharm
- Now, let's open up Notepad (or any similar program in Mac/Linux)...

Writing and running a script

- First we write some commands in Notepad
- Note that Python is sensitive to indentation — where MATLAB uses "end" and R brackets, Python uses indentation to indicate nested statements
- Starting with something simple...

```
print(2+2)
```

- Save this using Notepad to a file called "test.py"
- Then, run the following line (for Windows)...

```
python3 C:\\your_directory_here\\test.py
```

- Or this line (for Mac/Linux)...

```
python3 /your_directory_here/test.py
```

Loops

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

- As previously mentioned, Python uses indentation to indicate nesting
- If we don't indent 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...
- You can also define your own custom objects to be 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
```

Nested loops

- Just like in other languages, we can nest for loops to do more powerful things

```
bases = [1, 2, 3]
expos = [1, 2, 3, 4]
for base in bases:
 print("Base:" + str(base))
 for expo in expos:
  print(str(base) + "^" + str(expo) + "=" + str(base**expo))
```

```
## Base:1
## 1^1=1
## 1^2=1
## 1^3=1
## 1^4=1
## Base:2
## 2^1=2
## 2^2=4
## 2^3=8
## 2^4=16
## Base:3
## 3^1=3
## 3^2=9
## 3^3=27
## 3^4=81
```

- Naturally, we can mix loops and other control statements

```
for i in range(10):
 if i < 3:
  print(i)
 elif i > 3 and i < 6:
  print(i*100)
 elif i == 8:
  break
 else:
  print("Hello")

## 0
## 1
## 2
## Hello
## 400
## 500
## Hello
## Hello
```

Making a spectrogram

Game Plan

1. Import one signal
2. Collect measurements of amplitude in each time frequency bin
3. Plot it
4. Save the plot to file
5. Loop the above four steps as necessary
6. Put everything in a script

Setting up the environment

- First, launch your Python terminal
- Then, we need to import the necessary modules

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


## Traceback (most recent call last):
##   File "<string>", line 5, in <module>
##   File "/usr/local/lib/python3.5/site-packages/sounc
##     _lib = _ffi.dlopen('portaudio')
##   File "/usr/local/lib/python3.5/site-packages/cffi/
##     lib, function_cache = _make_ffi_library(self, na
##   File "/usr/local/lib/python3.5/site-packages/cffi/
##     backendlib = _load_backend_lib(backend, libname,
##   File "/usr/local/lib/python3.5/site-packages/cffi/
##     raise OSError(msg)
```

Importing a sound file

- Run the following lines (may need to tweak the location to match where your file is located!)...

```
location =
"/home/daniel/Desktop/samples/m03ae.wav"
[fs, y] = wavfile.read(location)
```

- This will place the sampled waveform contained in m03ae.wav with sample rate fs into the numpy vector y
- We can see what this waveform looks like with ...

```
y = y/np.max(np.abs(y))
plt.plot(y)
```

- And what it sounds like with...

```
sd.play(y,fs)
```