Python in 10 minutes

Part 4:

Dr. Mark Williamson

Purpose:

- Quick, bite-size guides to basic usage and tasks in Python
- I’m no expert, I’ve just used it for various tasks, and it has made my life easier and allowed me to do things I couldn’t manually
- I’d like to share that working knowledge with you

Lesson 4: Splitting Data

 Last time, we learned how to open and examine large datasets that otherwise could not be fully examined in typical spreadsheet format (ex. Excel). Today, we'll continue the use of large datasets and discover how to split datasets into smaller, more manageable parts. First, we will break one large dataset into several, equal sized chunks. Second, we will pull out a subset of specific data from the same large dataset.

Lesson 4: The Datasets in Question

- Silva 132 Eukaryotic fasta file
  - Contains 18S ribosomal RNA sequences from Eukaryotes
  - Available alongside this presentation at https://med.und.edu/daccota/files/docs/berdc_docs/silva-132-eukaryotic.zip
  - Cleaned from larger dataset from https://www.arb-silva.de
- Fasta file: text-based format to represent DNA or amino acid sequence
  - Consists of a heading that starts with “>”
    - Often has a series of information
    - Ex. >Sequence1; assembly1;
  - Followed by sequence lines
    - Depending on size of DNA/amino acid sequence, can be one or many lines
    - Ex. CCCAATAGGCAGCCGTATACACCGGTTATATA

Lesson 4: Chunking Your Data

Goal: Split the large dataset into 4, smaller and equal sized datasets

Examining the Dataset

- Download dataset (SILVA_132_Eukaryotic.fasta) and unzip
- Open Python and start a new file
  - Create a **path** and **file** variable (see Python in 10: Part 3 for more information)
  - Create a **title_num** variable and set it equal to 0
- Create a for-loop for each line
- Create an if statement that will add 1 to the **title_num** variable if there is an “>”
  - this will give us the total number of sequences rather than lines
  - It only counts when the line is a title
- Print out **title_num**, as well as **title_num/4**
  - Should be 77541 and 19385.25
  - We’ll round up to 19386 to use as our cutoff

```
#Checking out files
path="F:\\SCS_2017\\Pooled_Zotu\\"
file="SILVA_132_Eukaryotic.fasta"
tally=0

#Chunking
title_num=0
for line in open(path+file):
    if ">" in line:
        title_num+=1
print(title_num)
print(title_num/4)
```

Lesson 4: Chunking Your Data 2

Creating outfiles and variables

- Create four output files (outfiles)
  - Silva_1-4 making sure to include “w” to tell Python that you are writing to the files
- Create a file list called **file_list** that contains each of those files
- Create variables called **file_num** and **title_num** and set them to zero

```
outfile1=open(path+"Silva_1.fasta","w")
outfile2=open(path+"Silva_2.fasta","w")
outfile3=open(path+"Silva_3.fasta","w")
outfile4=open(path+"Silva_4.fasta","w")
file_list=[outfile1, outfile2, outfile3, outfile4]
file_num=0

title_num=0
```

Lesson 4: Chunking Your Data 3

Splitting the Dataset

- Create a for-loop for each line
- Create an if statement that checks for “>” in the line and adds 1 to **title_num**
- Create another if statement that check if **title_num** is equal to 19386
  - Within the if statement, reset **title_num** to 0 , add 1 **file_num**, and print a string to tell you your progress
  - Make sure to use == and not =
- After the two if statements, write the line to the current file from the file list
- Outside the for-loop, print a progress string, then close all four outfiles
- You should be able to open and view the resulting fasta files in Excel

Summary:

- **our code counts titles and writes lines to the first outfile until it reaches the cutoff value (1/4 of the total title number)**
- **then it resets the count, switches to the next outfile, and continues**
- **in the end, there are four outfiles produced, each about the same size**
- **the 4th outfile has one less title number**

```
for line in open(path+file):
    if ">" in line:
        title_num+=1
    if title_num==19386:
        title_num=0
        file_num+=1
        print("File Done: " + str(file_num))
    file_list[file_num].write(line)
print("File Done: 4")

outfile1.close()
outfile2.close()
outfile3.close()
outfile4.close()
```

Lesson 4: Sub-setting Your Data by Criteria

Goal: Create a subset of the data based on criteria from the title

Subset title only

- Create another output file
  - Call it Silva_Chordata as we'll subset only eukaryotes that are chordates
  - Chordates are animals with a notochord (vertebrates, tunicates, and lancets)
- Create a for-loop for each line
- Create an if statement that will write the line to the outfile if the string "Chordata" is present in the line
  - Titles without "Chordata" in them will not be written to the outfile
  - This will only give titles, not the sequences
- Outside the for-loop, close the outfile and print a status string

```
#Subsetting
outfile5=open(path+"Silva_Chordata.fasta","w")

for line in open(path+file):
    if "Chordata" in line:
        outfile5.write(line)
outfile5.close()
print("Chordata file done")
```

Lesson 4: Sub-setting your Data by Criteria 2

Subset title and sequence

- Create a final output file
  - Include “full” in the title to distinguish it from our earlier outfile with only the titles of the Chordates
- Create variable called **Chord_switch** and set it to False
  - This will act as a ‘switch’ in our later for-loop
  - If the switch is on, we’ll write to our outfile
  - If the switch is off, we won’t write
- Create a for-loop for each line
  - Create an if statement that checks if “>” is in the line
  - Create an if-else statement within the first if statement
  - If “Chordata” present, set **Chord_switch** to True
  - Else, set **Chord_switch** to False
  - Create another if statement (outside the first if-statement) that will write the line to the outfile if Chord_switch is True
- Outside the for-loop, close the outfile and print a status string

```
outfile6=open(path+"Silva_Chordata_full.fasta","w")
Chord_switch=False
for line in open(path+file):
    if ">" in line:
        if "Chordata" in line:
            Chord_switch=True
        else:
            Chord_switch=False
    if Chord_switch: #equivalent to 'if Chord_switch==True:'
        outfile6.write(line)
outfile6.close()
print("Chordata full file done")
```

Lesson 4: Summary

- Python can quickly split or subset large datasets
- Smaller chunks or subsets can be written to new files
- To do so, you can use for-loops, if-else statements, and other functions and methods to aid you
- This is often very useful for genetic data (fasta files)