Python, MACS, UCSC, BedTools

Zack McCaw

March 1, 2016

Python Resources

- Anaconda distribution of Python: https://www.continuum.io/downloads
- Example Python tutorials:
  - Code academy: https://www.codecademy.com/
  - Python tutorial: https://docs.python.org/2/tutorial/index.html

Python Flow

- Lists
  - Comma separated items delimited by square brackets
- For loops
  - For loop iterates over items in a list
  - Commands within the for statement are tabbed
- Conditionals
  - Executes tabbed commands if argument evaluates to true

```
# Create list of integers
A = [(i) for i in range(0,11)];

# Print list
for n in A:
    print("Number %d" % n);

# Print evens
for n in A:
    if (n%2==0):
        print(n)
```

Python Lists

- Lists are indexed starting at 0
- $A[n_1 : n_2]$ returns items n_1 up to, but not including, n_2
- Remove item at position j: `A.pop(j)`
- Insert x at position j: `A.inert(j,x)`

```
# Lists
A = [0,1,2];
B = ["a","b","c"];
# Concatenate
C = A + B;
# Slice
C[2:4]
# Length
len(C)
```

Python Functions

- Function definition
  - Syntax: `def` function name (arguments):
  - `return` halts execution and returns its argument
- Modules
  - To load functions from a module:
    `import [module name]`
  - re: regular expressions
  - os: operating system
  - numpy: matrix operations

```
# Define Fibonacci function
def fib(n):
    if(n==0 or n==1):
        return(1)
    else:
        out = fib(n-1) + fib(n-2);
    return(out)

# Evaluate
fib(13)
```

Python Strings

- String functions use the syntax: `[string].function()`
- `.split([delim])` will split on a delimiter, space is default
- `[str].join([list])` collapses the elements of a list into a string, separated by `[str]`
- `m = re.search([reg],[str])` searches a string using a regular expression.
  - If parentheses are used to capture a substring, this is accessed using `m.group(1)`

```
import re

# String
A = 'gene_name "TNF" ';
# Remove padding spaces
B = A.strip();
# Get first word
m1 = re.search("([^\s]+)",B);
# Gene gene name
m2 = re.search("\"(.*)\"",B);
```

Python File Processing

- Outline for looping over text file:
  - Open input file connection
  - Open output file connection
  - Loop over lines in the input file
  - Write lines to output file

```
## Read file line by line

# Input file
fin = open([filename],'r');
# Output file
fout = open([filename],'w');
# Loop
for line in iter(fin):
    # Split
    g = line.split();
    # Actions ...
    fout.write(output);
```

MACS: Remove Duplicates

- Model based analysis of ChIP-Seq (MACS) is a 'peak calling' method for identifying genomic binding sites from read count data
- Documentation: MACS
- Remove duplicates to mitigate amplification bias:
  - `-i` Intput file
  - `-g` Genome size
  - `--keep-dup` Duplicates to keep
  - `-o` Output file

```
macs2 filterdup -i In.bam -g mm --keep-dup 1 -o Out.bed
```

MACS: Sample Down

- Imbalance in read counts between treatment and control can bias results
- To sample down the condition that has more reads:
  - `-t` Intput file
  - `-n` Number to sample
  - `-o` Output file

```
macs2 randsample -t In.bed -n 10^6 -o Out.bed
```

MACS: Call Peaks

- Peak calling
  - `-t` Treatment file
  - `-c` Control file
  - `-f` File format
  - `-n` Output prefix

```
macs2 callpeak -t Trt.bed -c Ctrl.bed -f BED -g mm -n Out
```

Genome Browser

- UCSC Genome Browser: https://genome.ucsc.edu/
- Tool for visualizing genomic intervals
- Select relevant genome (species and assembly)
- Search for genes or genomic loci
- Upload intervals from bed file using add custom tracks

BED File Format

- Reference: BED Format
- Required fields:
  - Chromosome
  - Start position
  - End position
- Note that the start position is zero based, and start$=n_1$ to stop$=n_2$ represents positions $(n_1 + 1, n_2)$
- Some optional fields:
  - Name
  - Feature score (e.g. p value)
  - Strand $\{+, -\}$

Bedtools Functions

- Documentation Bedtools
- In subsequent examples, `File1.bed` is typically a genomic feature file, and `File2.bed` a read file
- Sort bed file:

```
bedtools sort -i File1.bed > Sorted.bed
```

- Intersect genomic intervals:

```
bedtools intersect -a File1.bed -b File2.bed > Out.txt
```

- Closest genomic interval:

```
bedtools closest -a File1.bed -b File2.bed > Out.txt
```

Bedtools Functions

- Annotate `File1.bed` with coverage by `File2.bed`

  ```
  bedtools annotate -i File1.bed -files File2.bed > Out.txt
  ```

- Convert `File1.bam` to `File1.bed`

  ```
  bedtools bamtobed -i File1.bam > File1.bed
  ```

- A genome file contains two fields specifying chromosome 1. name and 2. length. To generate random intervals from a genome:

  ```
  bedtools random -n 10^6 -g human.hg19.genome > Out.bed
  ```

- Shuffle locations of genomic intervals:

  ```
  bedtools shuffle -i File1.bed -g human.hg19.genome > Out.bed
  ```