Quiz Section Week 3
April 11, 2017

Today’s goals

- Stats review: p-values, null distributions, and significance testing
- Command line tips and tricks
- Python: Precedence, For loops and functions

P-values!

- P-values tell you about expectations *under the null hypothesis*
  - they say *nothing* about the alternative hypothesis or how probable it is

- Null hypothesis: usually the boring default, devil's advocate position – what you want to see if you can disprove

P-values!

- P-values tell you about expectations *under the null hypothesis*
  - they say *nothing* about the alternative hypothesis or how probable it is

- Null hypothesis: usually the boring default, devil's advocate position – what you want to see if you can disprove

There is *no difference* between treatment groups
Life expectancy is *not changing* over time
This coin is *not weighted*
These two sequences are *unrelated*

Historic example: R.A. Fisher and the tea-tasting test

8 cups of tea, randomly chosen to either have tea poured over milk or milk poured over tea

Null hypothesis?

How many would she guess correctly if she were picking randomly?

Null distribution: What we suppose the data might look like if the null hypothesis is true

- This could be based on a parameterized probability distribution
  - E.g. Poisson: number of successes in x tries with y% probability of success

- Or you can generate an *empirical* null based on your real data
  - E.g. Shuffle the labels of the variable you want to test

- Defining the most appropriate null distribution is a relevant and tough problem in a lot of computational biology research!

Multiple testing can be dangerous!

- http://fivethirtyeight.com/features/you-cant-trust-what-you-read-about-nutrition/
- Nutrition & lifestyle questionnaires from 54 individuals

Our shocking new study finds that ...

EATING OR DRINKING	IS LINKED TO
Raw tomatoes	Judaism
Egg rolls	Dog ownership
Energy drinks	Smoking
Potato chips	Higher score on SAT math vs. verbal
Soda	Weird rash in the past year
Shellfish	Right-handedness
Lemonade	Belief that "Crash" deserved to win best picture
Fried/breaded fish	Democratic Party affiliation
Beer	Frequent smoking
Coffee	Cat ownership
Table salt	Positive relationship with Internet service provider
Steak with fat trimmed	Lack of belief in a god
Iced tea	Belief that "Crash" didn't deserve to win best picture
Bananas	Higher score on SAT verbal vs. math
Cabbage	Innie bellybutton

SOURCE: FFQ & FIVETHIRTYEIGHT SUPPLEMENT

Multiple testing can be dangerous!

- http://fivethirtyeight.com/features/you-cant-trust-what-you-read-about-nutrition/
- Nutrition & lifestyle questionnaires from 54 individuals

Our shocking new study finds that ...

EATING OR DRINKING	IS LINKED TO	P-VALUE
Raw tomatoes	Judaism	<0.0001
Egg rolls	Dog ownership	<0.0001
Energy drinks	Smoking	<0.0001
Potato chips	Higher score on SAT math vs. verbal	0.0001
Soda	Weird rash in the past year	0.0002
Shellfish	Right-handedness	0.0002
Lemonade	Belief that "Crash" deserved to win best picture	0.0004
Fried/breaded fish	Democratic Party affiliation	0.0007
Beer	Frequent smoking	0.0013
Coffee	Cat ownership	0.0016
Table salt	Positive relationship with Internet service provider	0.0014
Steak with fat trimmed	Lack of belief in a god	0.0030
Iced tea	Belief that "Crash" didn't deserve to win best picture	0.0043
Bananas	Higher score on SAT verbal vs. math	0.0073
Cabbage	Innie bellybutton	0.0097

SOURCE: FFQ & FIVETHIRTYEIGHT SUPPLEMENT

Multiple testing can be dangerous!

- http://fivethirtyeight.com/features/you-cant-trust-what-you-read-about-nutrition/
- Nutrition & lifestyle questionnaires from 54 individuals

This is exactly the same as testing for alignments between thousands of sequences

Our shocking new study finds that ...

EATING OR DRINKING	IS LINKED TO	P-VALUE
Raw tomatoes	Judaism	<0.0001
Egg rolls	Dog ownership	<0.0001
Energy drinks	Smoking	<0.0001
Potato chips	Higher score on SAT math vs. verbal	0.0001
Soda	Weird rash in the past year	0.0002
Shellfish	Right-handedness	0.0002
Lemonade	Belief that "Crash" deserved to win best picture	0.0004
Fried/breaded fish	Democratic Party affiliation	0.0007
Beer	Frequent smoking	0.0013
Coffee	Cat ownership	0.0016
Table salt	Positive relationship with Internet service provider	0.0014
Steak with fat trimmed	Lack of belief in a god	0.0030
Iced tea	Belief that "Crash" didn't deserve to win best picture	0.0043
Bananas	Higher score on SAT verbal vs. math	0.0073
Cabbage	Innie bellybutton	0.0097

SOURCE: FFQ & FIVETHIRTYEIGHT SUPPLEMENT

Bonferroni correction: just raise the threshold depending on the total # of tests

- For 1000 tests: Use a threshold 1000x stricter
  - Does not require tests to have a particular relationship with each other
  - Ensures that the probability of rejecting a true null hypothesis is still less than your original desired p-value threshold
  - Suppose they did 1000 tests for this study
  - (50 lifestyle qs and 200 foods)
    - What's the corresponding E-value for p=0.0001?

Our shocking new study finds that ...

EATING OR DRINKING	IS LINKED TO	P-VALUE
Raw tomatoes	Judaism	<0.0001
Egg rolls	Dog ownership	<0.0001
Energy drinks	Smoking	<0.0001
Potato chips	Higher score on SAT math vs. verbal	0.0001
Soda	Weird rash in the past year	0.0002
Shellfish	Right-handedness	0.0002
Lemonade	Belief that "Crash" deserved to win best picture	0.0004
Fried/breaded fish	Democratic Party affiliation	0.0007
Beer	Frequent smoking	0.0013

- FYI: Sometimes this is too harsh, and *false discovery rate* corrections can be more useful

Programming

Note: If you are new to coding and still spending lots of time on lots of small errors , this is 100% normal!! Keep at it and it will get easier.

Useful command line tools

- **ls** []= list files
  - **ls –lh** = list files with extra detail
- **cd** []= change directory
  - **cd ..** = change to the parent directory
- **mkdir** []= make a new directory
- **rm** []= delete files
- **mv** []= move a file to a different name or location
- **cp** []= copy a file
- **pwd** []= print the current directory
- **head** []= print first lines of a file, **tail** []= print last lines of file
- **cat** []= print contents of a file
- **less** []= print contents of a file
- **grep** [] []= search for a string in a file

A pretty good starter reference: https://developer.apple.com/library/content/documentation/OpenSource/Conceptual/ShellScripting/CommandLInePrimer/CommandLine.html

- up arrow = previous command history
- tab = autocomplete file name
- | pipe = use output as input to next command

- You can install Python modules from the command line using **pip**
- Much much more…

Practice:

- Navigate to wherever you saved your homework 1 python script
- take a look at the first lines of it using “head”
- Use grep to search for the “%” operator
- then rename the file using “mv”

Order of operations/**precedence**

- Same rules apply as for mathematical expressions generally, plus some other conventions
- Without other specifications, evaluation happens left-to-right

What's the value of each expression?

```
2 ** 7 == 127 + 1
2 + 2 ** 7 == 128 + 1
2 + (2 ** 7 == 128)+1
2 == 2 or 3 == 2 == False
2 == 2 or 3 in [3,4] == False
(2 == 2 or 3 in [3,4]) == False
```

Reference: https://docs.python.org/2/reference/expressions.html#operator-precedence

Order of operations/**precedence**

- Same rules apply as for mathematical expressions generally, plus some other conventions
- Without other specifications, evaluation happens left-to-right

What's the value of each expression?

```
2 ** 7 == 127 + 1
2 + 2 ** 7 == 100
(2 + 2 ** 7 == 100)+1
2 == 2 or 3 == 2 == False
2 == 2 or 3 in [3,4] == False
(2 == 2 or 3 in [3,4]) == False
```

When in doubt: Use parentheses!!!

Reference: https://docs.python.org/2/reference/expressions.html#operator-precedence

For loops let you repeatedly apply the same lines of code to each element in a list

```
x = [1, 2, 3]
for i in x:
    print i
print 'done!'
```

i takes on the value of each element in the list for each iteration of the code inside the for loop block

For loops let you repeatedly apply the same lines of code to each element in a list

```
x = [1, 2, 3]
for i in x:
    print i
print 'done!'
```

```
1
2
3
done!
```

i takes on the value of each element in the list for each iteration of the code inside the for loop block

For loops also work for strings!

```
x = 'actg'
for i in x:
    print i
print 'done!'
```

```
a
c
t
g
done!
```

Compute the sum of the numbers in list x!

```
x = [1,2,4,5]
```

```
sum = 0
for v in x:
    sum = sum + v
print 'The sum is:', sum
```

```
The sum is:  12
```

How about the product?

```
x = [1,2,4,5]
```

```
print 'The product is:', product
```

```
The product is:  40
```

How about the product?

```
x = [1,2,4,5]
```

```
product = 1
for v in x:
    product = product * v
print 'The product is:', product
```

```
The product is:  40
```

Powerful strategy: Combining for loops and if/else statements

```
x = [12, 3, 4.4, 6]
```

Output how many numbers in the list x with values greater than 5

```
count = 0
for v in x:
    if v > 5:
        count = count + 1
print count
```

Watch the indents!!

Example: reverse complement

```
s = 'ATCG'

reverse_complement = ''
for nuc in s:
        # Find the complement of the nucleotide









        # Add the complement to the beginning of new string
        reverse_complement = nuc + reverse_complement
print reverse_complement
'CGAT'
```

Example: reverse complement

```
s = 'ATCG'

reverse_complement = ''
for nuc in s:
        # Find the complement of the nucleotide
        if nuc == 'A':
                nuc = 'T'
        elif nuc == 'T':
                nuc = 'A'
        elif nuc == 'C':
                nuc = 'G'
        elif nuc == 'G':
                nuc = 'C'
        # Add the complement to the beginning of new string
        reverse_complement = nuc + reverse_complement
print reverse_complement
'CGAT'
```

Functions are sub-programs that you can call in one line

```
>>> x = [1, 2, 3]
>>> print len(x)
3
```

- Used as a single word (no spaces) followed by “()”, where the input to the function goes within the parentheses
- The function will run, and it will evaluate to the output of the function

```
>>> print len( ‘hello!’ )
```

Functions are sub-programs that you can call in one line

- Python has many built-in functions (e.g. len)
- Modules contain definitions for additional functions
- Next week we will talk about defining and writing your own functions

Useful list functions

```
# Initializing a sequence of integers
x = range(0,4)
print x
[0, 1, 2, 3]

# Adding to the end of a list
x.append('four')
print x
[0, 1, 2, 3, four]
```

Iterate through a list with range() and indices

```
english = ['zero','one','two']
spanish = ['cero', 'uno', 'dos']

for i in range(0,len(english)):
     print english[i], spanish[i]
```

Iterate through a list with range() and indices

```
english = ['zero','one','two']
spanish = ['cero', 'uno', 'dos']

for i in range(0,len(english)):
     print english[i], spanish[i]
```

```
zero cero
one uno
two dos
```

Useful string functions

These are defined to manipulate a specific string variable, so we use the . to reflect that (more on this later)

```
>>> s = "GATTACA"
>>> s.find("ATT")
1
>>> s.count("T")
2
>>> s.lower()
'gattaca'
>>> s+s
'GATTACAGATTACA'
```

```
>>> s.upper()
'GATTACA'
>>> s.replace("G", "U")
'UATTACA'
>>> s.replace("C", "U")
'GATTAUA'
>>> s.replace("AT",
"**")
'G**TACA'
```

Extra practice:

https://interactivepython.org/runestone/static/thinkcspy/index.html

Sections 4 and 6