Hypothesis Testing, Project 2, and Google Earth

April 10, 2012

Today

- Hypothesis Testing
  - Coin Flipping
  - Word Frequencies
- Project 2
  - Wrapper Function
  - Test Functions
  - Error Handling
- Google Earth

Hypothesis Testing

Alternate hypothesis: The coin is unfair.

Null hypothesis: The coin is heads with prob. 0.5

- We want to *reject* the null hypothesis by showing that it is **unlikely** to get only 3 heads with a fair coin.

Prove that the coin is NOT innocent.

- Open coinflip.xls

Hypothesis Testing

Word frequencies:

1. Calculate the word frequencies of the stop words in the two books
2. Normalize the word frequencies

	a	able	about	across	after	...
File 1	.3	.01	.003	.0027	0.006	...
File 2	0.238	0.0932	0.0034	0.0021	0.05	...

Alternate hypothesis: Both books come from the **same** distribution of frequencies.

Null hypothesis: The books come from **different** distributions of frequencies.

Hypothesis Testing

	the	of	a	to
File 1	.3	.2	.45	.05
File 2	.1	.3	.5	.1

`compareTwo(list1,list2)`

gives me some distance x

Hypothesis Testing

	the	of	a	to
File 1	.3	.2	.45	.05
File 2	.1	.3	.5	.1

`compareTwo(list1,list2)`
gives me some distance x

	the	of	a	to
File 1	.2	.45	.3	.05
File 2	.1	.3	.1	.5

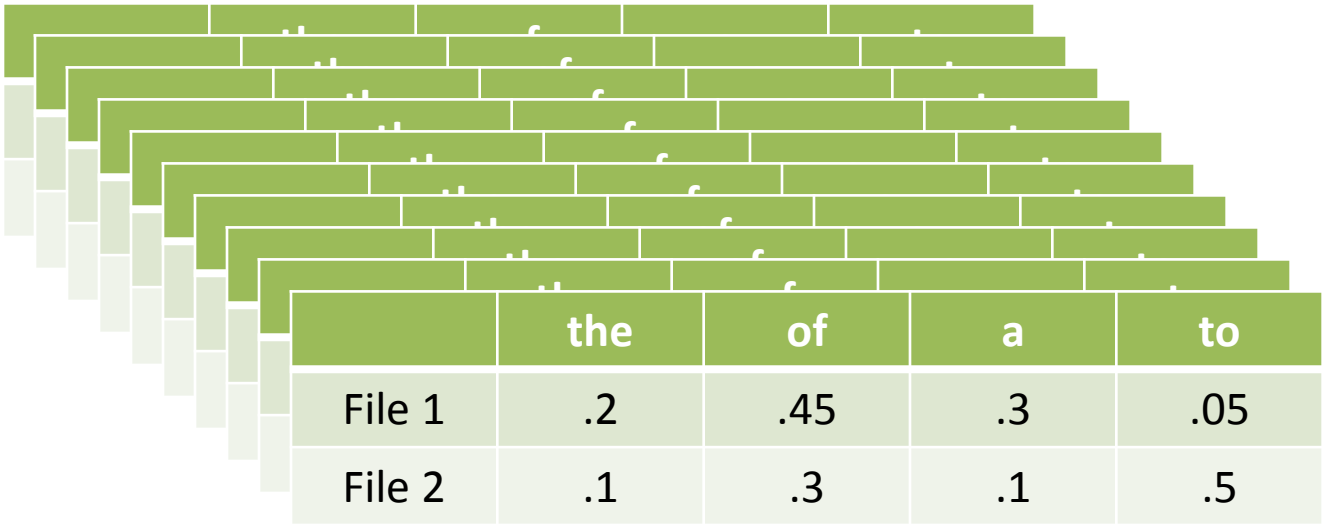
`compareTwo(list1,list2)`
gives me some distance y

Randomly shuffle each list.

Hypothesis Testing

	the	of	a	to
File 1	.3	.2	.45	.05
File 2	.1	.3	.5	.1

`compareTwo(list1,list2)`
gives me some distance x

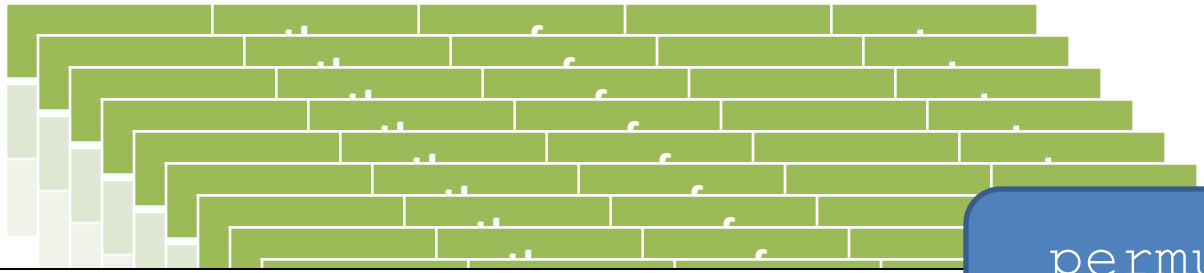


Do this 1,000 times: `compareTwo(list1,list2)`
gives me distances y_1 y_2 y_3 $y_{4...}$ y_{1000}

Hypothesis Testing

	the	of	a	to
File 1	.3	.2	.45	.05
File 2	.1	.3	.5	.1

`compareTwo(list1,list2)`
gives me some distance x



`permutationTest.py`
(online)

```
>>> runPermutationTest()
lists have a distance of 0.1
is that significant? Run permutation test 1000 times.
0.121 % of the tests were at least as similar as orig
list.
p-value is not statistically significant
```

Today

- Hypothesis Testing
  - Coin Flipping
  - Word Frequencies
- Project 2
  - Wrapper Function
  - Test Functions
  - Error Handling
- Google Earth

Example Skeleton Code

```
def runExperiments(list1,list2,origval,tot):
    '''Runs permutation tot times.
    INPUTS: ... OUTPUTS: ...'''
    return
def permute(list1,list2):
    '''Randomly shuffles lists and returns compareTwo()
    INPUTS: ... OUTPUT: ... '''
    return
def compareTwo(list1,list2):
    '''Returns the avg sum of their differences (auth. act)
    INPUTS: ... OUTPUTS: ....'''
    return
```

Python Wrapper Function

```
def runExperiments(list1,list2,origval,tot):
    '''Runs permutation tot times.
    INPUTS: ... OUTPUTS: ...'''
    return
def permute(list1,list2):
    '''Randomly shuffles lists and returns compareTwo()
    INPUTS: ... OUTPUT: ... '''
```

```
def runPermutationTest():
    '''Runs the permutation test on the
two lists and prints the significance.
    INPUTS: none
    OUTPUTS: none'''
    return
```

Test Functions

```
def testPermutation():
    '''tests permutation. INPUTS: none OUTPUTS: none'''
    list1 = [0,0,1,0,0]
    print 'list1:',list1
    list2 = [0,0,1,0,0]
    print 'list2:',list2

    origval = compareTwo(list1,list2)
    print 'Orig val is',origval,'(I expect it to be 0.0)'

    print 'I expect the prob of this happening is 5/25 = 0.2'
    perc = runExperiments(list1,list2,origval,1000)
    print perc,'% when calling runExperiments()`
    return
```

Test Functions

```
def testPermutation():
    '''tests permutation. INPUTS: none OUTPUTS: none'''
    list1 = [0,0,1,0,0]
    print 'list1:',list1
    list2 = [0,0,1,0,0]
    print 'list2:',list2

    origval = compareTwo(list1,list2)
    print 'Orig val is',origval,'(I expect it to be 0.0)'
```

```
>>> testPermutation()
list1: [0, 0, 1, 0, 0]
list2: [0, 0, 1, 0, 0]
Orig val is 0.0 (I expect it to be 0.0)
I expect the prob of this happening is 5/25 = 0.2
0.213 % when calling runExperiments()
```

Error Handling

```
...
if len(list1) != len(list2):
        print 'ERROR! lists are not the same length!'
        return -1
...
```

Error Handling

```
...
if len(list1) != len(list2):
        print 'ERROR! lists are not the same length!'
        return -1
...
```

```
...
myMatch = re.search('\*\*\*\s*START.+\n',myString)
if myMatch == None:
        print 'Error! \'***\\s+START.+\\n\' does not exist.'
        return
...
```

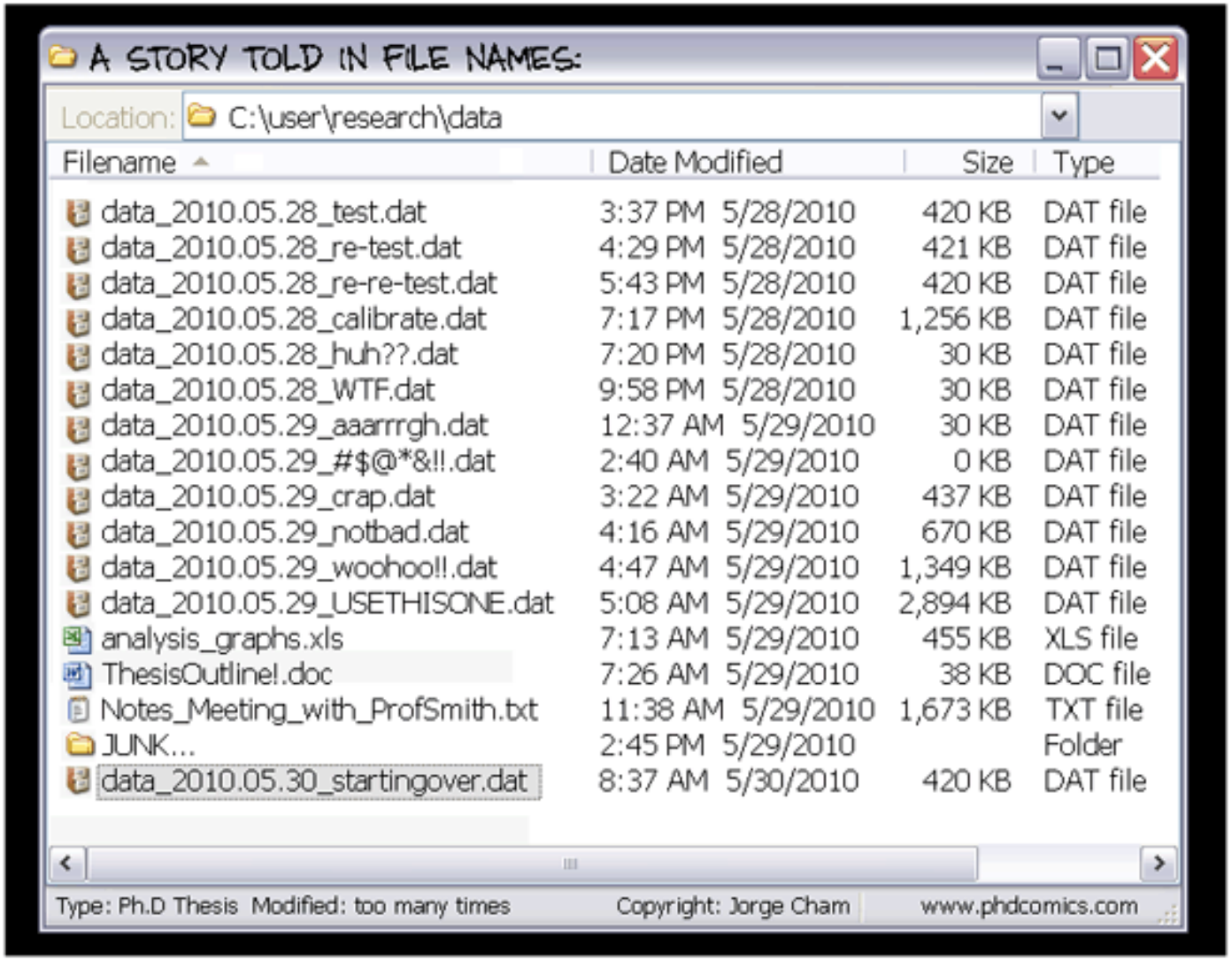
Project 2 Deadlines

- Feedback over the weekend
  - Not going to be as in-depth as the first project

Sun	Mon	Tues	Wed	Thurs	Fri	Sat
4/1	4/2	4/3	4/4	4/5	4/6	4/7
				Proposal Due		
4/8	4/9	4/10	4/11	4/12	4/13	4/14
				Project 2 Due (Code)		
4/15	4/16	4/17	4/18	4/19	4/20	4/21
		Project 2 Due (Website)				

Break

Download Google Earth (link on website) if on a laptop.



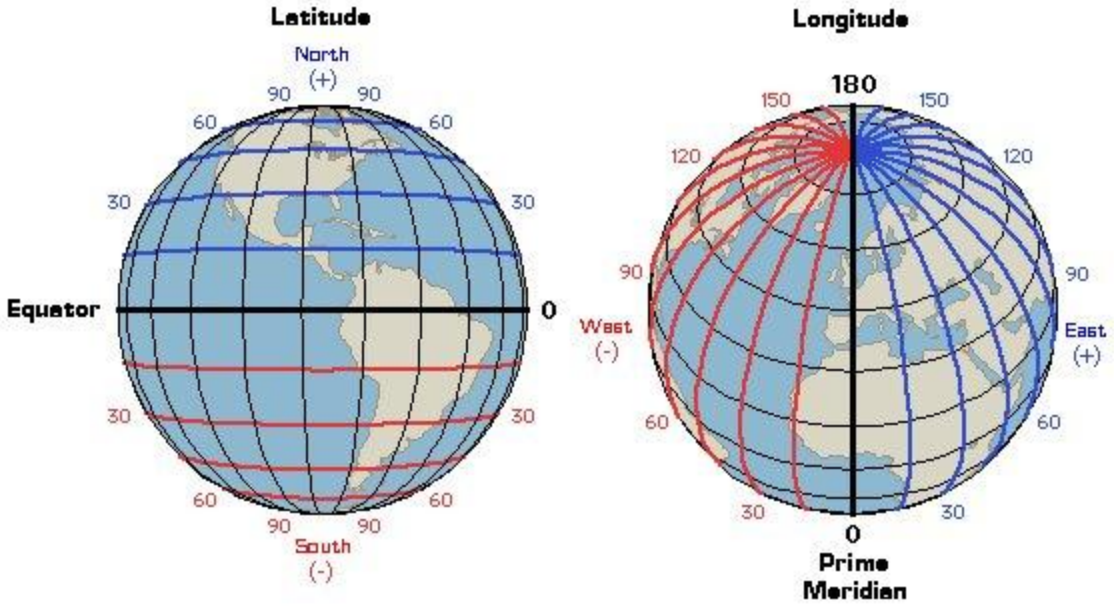
http://www.phdcomics.com/comics/archive.php?comicid=1323

Today

- Hypothesis Testing
  - Coin Flipping
  - Word Frequencies
- Project 2
  - Wrapper Function
  - Test Functions
  - Error Handling
- Google Earth

CIT.kml

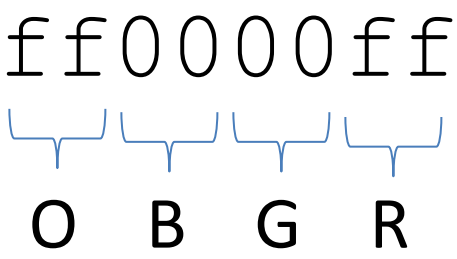
- Try changing the following things:
  - Name
  - Description
  - Size of Pin
  - Coordinates
  - What about color?



http://geographyworldonline.com/tutorial/instructions.html

KML Color Codes

O = Opacity
B = Blue
G = Green
R = Red



These are two-digit *hexadecimal* numbers.

Two-Digit Numbers

Decimal (10 digits): 0-9

00 01 02 03 04 05 06 07 08 09 ?

Two-Digit Numbers

Decimal (10 digits): 0-9

00 01 02 03 04 05 06 07 08 09 10 11 12 ... 99

Two-Digit Numbers

Decimal (10 digits): 0-9

00 01 02 03 04 05 06 07 08 09 10 11 12 ... 99

Binary (2 digits): 0-1

00 01 ?

Two-Digit Numbers

Decimal (10 digits): 0-9

00 01 02 03 04 05 06 07 08 09 10 11 12 ... 99

Binary (2 digits): 0-1

00 01 10 11

Two-Digit Numbers

Decimal (10 digits): 0-9

00 01 02 03 04 05 06 07 08 09 10 11 12 ... 99

Binary (2 digits): 0-1

00 01 10 11

Hexadecimal (16 digits): 0-9,a-f

00 01 ... 08 09 0a 0b 0c 0d 0e 0f ?

Two-Digit Numbers

Decimal (10 digits): 0-9

00 01 02 03 04 05 06 07 08 09 10 11 12 ... 99

Binary (2 digits): 0-1

00 01 10 11

Hexadecimal (16 digits): 0-9,a-f

00 01 ... 08 09 0a 0b 0c 0d 0e 0f 10 11 ... fe ff

Two-Digit Numbers

Decimal (10 digits): 0-9

00 01 02 03 04 05 06 07 08 09 10 11 12 ... 99

Binary (2 digits): 0-1

00 01 10 11

Hexadecimal (16 digits): 0-9,a-f

00 01 ... 08 09 0a 0b 0c 0d 0e 0f 10 11 ... fe ff

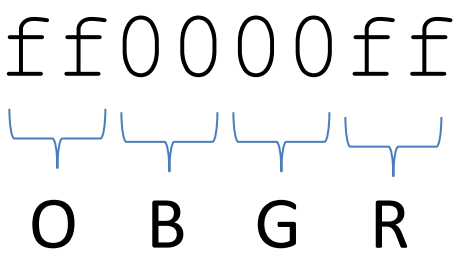
Decimal: $10^2 = 100$ two-digit #s

Binary: $2^2 = 4$ two-digit #s

Hexadecimal: $16^2 = 256$ two-digit #s

KML Color Codes

O = Opacity
B = Blue
G = Green
R = Red



Hexadecimal (16 digits): 0-9,a-f

00 01 ... 08 09 0a 0b 0c 0d 0e 0f 10 11 ... fe ff