CS 696 Intro to Big Data
Fall Semester, 2016
Assignment 2

San Diego State University -- This page last updated 9/13/16

Due Sept 22 23:59

The goal of this assignment is to use map, reduce, and filter to solve the problems. You should not need to use for or while statements. The solutions with map, reduce and/or filter may not be as efficient as solutions using a for or while. However the core operations in Hadoop are map and reduce. While there is a difference in the Hadoop mapReduce from what you will use hear it is important to get used to using those constructs. Don't forget that one can replace filter with reduce.

If your functions use different names the test cases will fail and you will lose points.

In each of the solutions to the problems you are to include tests that show that your code works. See problem 2 & 3 for examples. Note that the examples given do not include all cases that you should test.

1. If we list all the natural numbers below 20 that are multiples of 3 or 5 but not multiples both of 3 and 5, we get 3, 5, 6, 9, 10, 12, 18. The sum of these multiples is 63. Write a function, sum_multiples_3_5, that returns the sum of the multiples of 3 and 5 but not both less than N.

2. Write a function, pattern_count with two arguments. The first arguments is a string, lets call it text, and the second argument is also a string, call it pattern. The function pattern_count returns the number of times the pattern occurs in the text. For example

```
using Base.Test

@test pattern_count("abababa", "aba") == 3
@test pattern_count("aaaaa", "aa") == 4
@test pattern_count("Abcde", "abc") == 0
```

3. A palindrome is a string that reads the same forwards and backwards. We will call a number octaldrome if its octal representation is a palindrome. So 945 is an octaldrome as in octal it is 1661. The first nine positive octaldromes are 1, 2, 3, 4, 5, 6, 7, 9 and 18. We want to find all octaldromes less than or equal to N. Write a function octaldromes(n) that returns all the octaldromes equal to or less than n. So we have

```
using Base.Test
@test octaldromes(1) == [1]
@test octaldromes(9) == [1,2,3,4,5,6,7,9]
```

4. Write a function, most_frequent_word, which has two arguments. The first argument is a string, the second argument is an integer, call it n. most_frequent_word returns the sequence word(s) of length n that occurs most in the string. For example

most_frequent_word(“TCGAAGCTAGACGCTAGTAGCTAGTGTGCA” 4) returns [“CTAG”, “GCTA”]

5. In DNA strings, symbols 'A' and 'T' are complements of each other, as are 'C' and 'G'. The reverse complement of a DNA string s is the string formed by reversing s, then taking the complement of each symbol (e.g., the reverse complement of "GTCA" is "TGAC"). Write a function reverse_complement(s) that returns the reverse complement of s.

6. The GC-content of a DNA string is given by the percentage of characters in the string that are 'C' or 'G'. For example, the GC-content of "AGCTATAG" is 37.5%. Write a function gc_content(s) with input a DNA-string and returns the GC-content of the string. So we have gc_content("AGCTATAG") == 0.375

7. Write a function digit_distribution with on argument an array of numbers (integers or floats) and returns a dictionary of the distribution of digits in the array of numbers. So with an input of [112, 24, 15] digit_distribution will return Dict(1 => 3, 2 => 2, 4 => 1, 5 => 1)

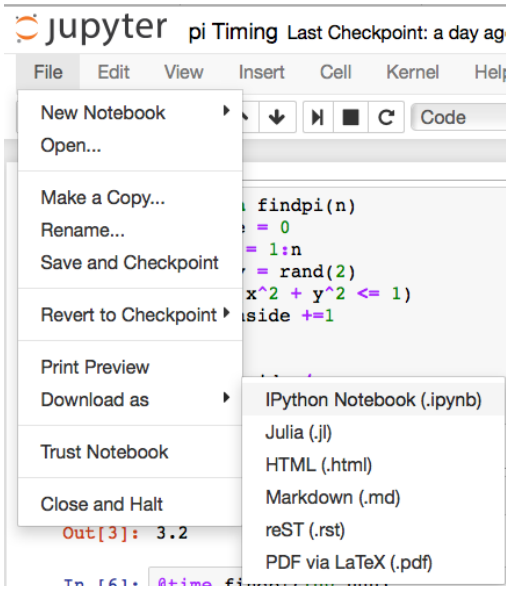
Grading

Each problem is worth 5 points. Four of the points are for solutions using map, reduce, and/or filter that produce the correct output. Solutions using map & reduce are highly prefered. Solutions that do not use map. reduce, and/or filterOne point is for style. That is did you use proper indentation, naming convention, good names and reasonable Julia constructs. Providing good tests for the problems is worth another 10 points total.

What to turn in

You are free to use any IDE to write your code. Normally I use Juno/Atom to develop and debug then move the code to Jupyter as autocomplete is better in Juno. However you are to turn in a Jupyter IJulia notebook containing the answers to the questions above. Since Jupyter notebooks can contain text and code, before each problem indicate which problem it is in text, not in code comment.

To turn in your assignment download your Jupyter notebook as an IPython Notebook (.ipynb). See image below. This will allow me to run your assignment in Jupyter. Do not download it as a Julia file (.ji) as this will not run in Jupyter and removes all the text (markdown). Note that when you download your assignment it will create a file with the extension .ipynb.json. I will remove the .json extension.



```
findpi(n)
= 0
= 1:n
= rand(2)
x^2 + y^2 <= 1)
side +=1
Out[3]: 3.2
```

Once you have downloaded the assignment zip it up and then upload the zip file to the course portal.

If you do not have a a program that can create zip files you can do it using Julia. Here is a program that will create a zip file.

```
using ZipFile
location = "Full/Path/To/Directory/Containing/Your/File"
file  = "assignment1.ipynb.json"
cd(location)
w = ZipFile.Writer("assignment1.zip");
f = ZipFile.addfile(w, file);
close(w)
```

Late Penalty

An assignment turned in 1-7 days late, will lose 5% of the total value of the assignment per day late. The eight day late the penalty will be 40% of the assignment, the ninth day late the penalty will be 60%, after the ninth day late the penalty will be 90%. Once a solution to an assignment has been posted or discussed in class, the assignment will no longer be accepted. Late penalties are always rounded up to the next integer value.