

Data analysis using python - Exercises

Student and Staff IT Introduction

April 16, 2013

1 Fibonacci sequence

The Fibonacci sequence is defined as:

$F_n = F_{n-1} + F_{n-2}$ where $F_0 = 0$ and $F_1 = 1$

So the first elements are:

0, 1, 1, 2, 3, 5, 8, 13, 21, ...

Create a python function having a parameter "n" computing the n^{th} element of this sequence.

2 Sequence analysis

The *fasta* file "sequences.fas" contains several genetic sequences. The fasta format looks like:

```
>Name of gene 1
ATGCGGCAGCATGCATGCATGCTAGCTAGTCAGTGTGTGATGCATG...
>Name of gene 2
ATGATAGTAGTGCAGTCAGTCGATGCATGCATGCATGCTAGCTAGT...
...
```

Each entry is composed by one unique line starting by '>' corresponding to the title of the sequence. The lines following the title describe the sequence. The sequence can be written on many lines.

Build a python script reading the fasta file and building a tabulation delimited file containing, for each sequence, the number of each nucleotide (A, T, G and C) in the sequence and the percentage of G+C nucleotides. The paths to the fasta file and the result file should be given as command line parameters.

3 Simple statistics

The csv file "gene_expression.csv" contains information about expression level for some genes. Each line corresponds to a different gene, the columns to some experiments. Fields are separated by commas. The first row contains the experiment names and the first column the gene names. You may want to use the following terminal command to display the content of the file:

```
column -s ',' -t -n < gene_expression.csv | less -S
```

Build a python script reading this csv file "gene_expression.csv", and computing the mean, the standard deviation of each column and the median for each gene. Create a new file with three more columns containing these values.

```

=====
# EXECISE 1 =====
# =====

#!/usr/bin/env python
# The previous line is a way to tell to your system where
# python interpreter is. Must be the very first line of the file

def fibonacci(n):
    if type(n) != int or n < 0:
        print "n should be a positive number"
        exit(2)
    if n == 0: return 0
    if n == 1: return 1
    a = 0 #F0 = 0
    b = 1 #F1 = 1
    print "fibonacci(0) = %d" % a
    print "fibonacci(1) = %d" % b
    for i in range(2,n+1):
        f = a + b #Fn
        a = b #Fn-2
        b = f #Fn-1
        print "fibonacci(%d) = %d" % (i,f)
    return f

# Main code
print fibonacci(7)

=====
# EXECISE 1 Version2 =====
# =====
# answer to the exercise 1 with a different method
# this is called a recursive method (i.e. the function calls itself)

def fibonnaci(n):
    if (n==0): # case 0 must be defined
        return 0
    elif (n==1): # case 1 must be defined
        return 1
    else:
        # any other case can be computed recursively
        # the function calls itself for a smaller n
        # until the defined case 0 and 1 are reached
        return fibonnaci(n-1) + fibonnaci(n-2)

# Main code
print fibonacci(7)

=====
# EXECISE 2 =====
# =====

#!/usr/bin/env python
import os
import sys

def read_sequence_and_compute_stats(seq):
    a = seq.count("A") #Number of A
    t = seq.count("T") #Number of T
    g = seq.count("G") #Number of G
    c = seq.count("C") #Number of C
    gc = (g+c)/float(len(seq))*100 # G+C percent
    return "%s\t%s\t%s\t%s\t%.2f" % (a,t,g,c,gc)

def read_file_and_find_sequences(fasta, result_file):
    if not os.path.isfile(fasta):
        print "Error: impossible to find the fasta file"
        exit(2) # exits the scripts and returns the error code 2
    fas = open(fasta, 'r')
    res = open(result_file, 'w')
    res.write("Sequence\tA\tT\tG\tC\tGC\n") # header of the file

```

```

title = ""
sequence = ""
for line in fas:
    line = line.strip() #remove \n at the end of the line
    if line.startswith(">"): # it is a title
        if len(title) != 0 and len(sequence) != 0: # there is already one sequence read
            stats = read_sequence_and_compute_stats(sequence) # compute the "stats" of the sequence
            res.write("%s\t%s\n" % (title, stats)) # write the result in the result file
            title = line[1:] # we don't want the ">" sign at the beginning of the title
            sequence = ""
        sequence += line

# Main code
if len(sys.argv) != 3:
    print "Error: the this command should have two parameters: the fasta file path and the result file path"
    exit(2) #Exit the script and return the error code '2'
else:
    read_file_and_find_sequences(sys.argv[1],sys.argv[2])

=====
# EXECISE 3 =====
# =====

#!/usr/bin/env python
import numpy as np
import sys

if len(sys.argv) != 3:
    print "Please give an input file and an output file"
    exit(2)

infile = sys.argv[1]
outfile = sys.argv[2]
f = open(infile, 'r') #input file
out = open(outfile, 'w') #output file
i = 0 # to count the lines in input file
for line in f:
    i+=1
    line = line.strip() #remove the \n sign at the end of the line
    if i == 1: # no computation to do on first line
        out.write("%s,std,mean,median\n" % line)
    else:
        first_comma_index = line.index(",")
        gene_name = line[:first_comma_index] # the gene name is the first element of the line
        line = line[first_comma_index+1:] # the gene name of the line should be remove of the line to have only the numeric data
        line_data = np.fromstring(line, dtype=float, sep=',') #build a numpy array from the string
        std = np.std(line_data) #standard deviation
        avg = np.mean(line_data) #mean
        median = np.median(line_data) #median
        out.write("%s,%s,%s,%s,%s\n" % (gene_name, line, std, avg, median)) #write the values in file

#close the files
f.close()
out.close()

```