PERL PROJECT

WORKING with GFF RECORDS

Extracting GFF Features from Sequences

Fernando Pozo Ocampo
fernando.pozo01@estudiant.upf.edu
December, 2016

—

Introduction: What are we looking for?

This report describes a program which works with GFF records and a simple FASTA file. Basically, It retrieves those substring from a DNA FASTA sequence that are delimited by the start and end coordinates of a GFF record.

It runs in a right way with most of GFF annotation reports reading the columns: seqname, source, feature, start, end, score, frame and attribute, separated by tabs. It obtains an output like the next one below:

CDS.NM_142033 5996 6911 GC(%)-> 46.943231441048 ATGGGCGTGGATCAGTCATT
exon.NM_142033 5996 6911 GC(%)-> 46.943231441048 ATGGGCGTGGATCAGTCATT
CDS.NM_142033 7209 7662 GC(%)-> 47.577092511013 ATGATGATGAACTTGAGAAT

printing...
seqname start end gccontent sequence

from...
seqname source feature start end score frame attribute

and calculates the *GC content* from sequence

Therefore, it also provides some useful functions, like a nucleotides counter or a direct chain `(5'->3')` to inverse `(3'<-5')` (reverse complement) converter. Four subroutines have been implemented in a local environment and it could be useful for another programs. My `for loop` and subroutine which parse GFF have been performed in a global environment in my script. The `for loop` includes the main functions of my program. Therefore, it calls another subroutines here. Talking about the second one, GFF parser have 4 different arrays, each of one is used for print the type of each feature, the coordinates (start and end), the GC content and the subsequent sequences of nucleotides from each one.

I have created a Perl script with approximately 125 lines of code trying to make my program easy to read and useful at the same time. In methods, I discussed how I have created this code, explained it in more detail and how I would improve it. In my discussion, I will add some comments related to solve several problems during my work, alternatives routes to manage it and also about possible applications of my code in bioinformatics.

Methods: How has the code been implemented ?

First of all, I have to say that my code have a main `for` statement as you can see in first lines. I call my subroutines at first line and after, then you read the main loop. Therefore, the logical way to describe this is

talk about the subroutines which parse the GFF and FASTA file, the next subroutines and finally describe my main for loop statement.

I wrote my subroutine "read_fasta" which parse my FASTA file (second argument in command line) creating a while loop statement which check if the first lines contains a greater than. Then, you can assume that you are reading a FASTA file. After this mark, all the subsequent file must be characters of nucleotides. However, I found an error with unitialized variables when I open my FASTA file with the while statement and the substring function. Then, immediately I have to use an unless statement in order to introduce my whole sequence in a variable `($dna)`. Also, you could add another regular expressions aiming to discard blank spaces or discard comment lines. In this case, our FASTA files performs results in a proper way with my code. It performs like an independent block.

Subroutine "read_gff" have global variables instead of local ones. I make it in this way because when I create four different arrays, one for each column of my output, and it seemed easier to implement in my script. I suppose you can work with local variables in this subroutine, if you work with hashes instead of arrays. The performance of the function follow the way adding one or more values to the end of each array with the aim of work with these ones in the main loop. Before, each of ones have been splitted with an space (tab). I separated with a point the features and groups, and deleted quotation marks and semicolon with a transliteration regular expression, supposing that it could not be the most elegant way to do it but... it works!

The GFF file has several columns with "-" strand. It is meaning that you have the coordinates in inverse chain of DNA. Then, I add a subroutine which extract the right nucleotides of the coordinates. The function `revcom` takes the first index of the array ($_[0] and initialize a variable ($dnago). In my case, it takes the each nucleotides sequence in direct change and works with it reversing the string. Immediately, it transliterate every nucleotides looking its complement. Here, you can also work with `@_` in the same way. The last variable, would take all the elements of the subsequent array.

Finally, it adds a counter of the proportion of GC nucleotides as a complementary work. It creates a function with a substring which counts one by one all the nucleotides of every simple chain. It takes in SWITCH all of every count and calculate the sum of G+C and its percentage. Due to my correction, I have to start my `for` statement by -1 position. This loop goes across the length of my sequence and take it one by one until its last nucleotide. I can print in my final output either the count or the percentage. In my case, I obtain the percentage of total GC content and the same for every feature.

Otherwise, my main loop was be implemented after to write that subroutines. After adding the function which control the number of arguments to run the program, it calls `read_gff` and `read_fasta`. It loads this functions taking all the values of `read_gff` (it was not created in a local environment as I said) and declaring a new variable for the return value of `read_fasta`. Then, loop statement start from 0 to `@names` in a scalar context one by one. The next step of my script was making the logical corrections which were exposed in the exercise description. It obtains a way to do the right substring function in string I loaded in `read_fasta`. Lastly, into my loop it adds an `if-elsif` condition for + and - strains, which logically, have to print different outputs from my sequence (direct and inverse chain). However, it is not necessary to compute the GC content after making the change, because of this content is the same in any case. It is important to say that it introduces `revcom` and `GC_counter` functions into the loop.

This script have been commented in almost every line in order to improve its comprehension.

Discussion: Troubleshooting and applications of my script

During development of my script, I tried different ways to solve my problems. First of all, I needed to parse all the GFF file in order to separate the main columns which I was interested in. At first, I thought it would be performed if I created a hash of list. Then, I had to print it, supposing that we found it easier in my way than in that way. Once I decided to make my four arrays, I divided them in `5'->3'` and `'3<-5'` chains in an

`if-elsif` statement. In the main loop, it also adds an optional `if` statement if you do not want to print all the features. In this case, with only 3 of them I chose to print all.

Parsing a FASTA file could also have several ways to do. In my case, I had a problem with the first value of my string. At first time, it was uninitialized and I had to say the program how it must include the string into a variable into my while statement. Another way to do it (with more lines of code and, maybe in a complex form), is available at `Beginning Perl for Bioinformatics`.

Additionally, I could add a nucleotide counter in order to find some tools to find biological. However, I had an extra problem with counter. In my introduction I commented that I had to start my `for` statement by -1 and it is due to the sequence corrections. I suppose there is a better way to solve it before this part of code, or maybe, it was something wrong when it parse the sequence in main loop. However, my output was the expected and I do not try to make in another different way, with only little changes I can not solve this problem.

One way to improve this script could be implement several different arguments to run it from command line (`-r, -s, -a for example`). For instance, if you does not want to see all features sequence, or you want to know how many GC nucleotides there are in each of them instead of percentages.

GFF is a standard file format for storing genomic features in a text file. It is usually used as a report format for visualize all the features in a simple output without include the whole sequence. This script could be useful for an user who wants to visualize the sequence for every features or simply for each one. Also, the GC content can be an useful tool in a biological context. This content is found to be variable with different organisms, the process of which is envisaged to be contributed to by variation in selection, mutational bias or biased recombination-associated DNA repair.