Deduper Part 1

Fall 2017, Bi624 - Amanda Leonti

Deduper Part 1 - Pseudocode

Instructions

Given a SAM file of uniquely mapped reads, remove all PCR duplicates (retain only a single copy of each read)

- Samtools sort
- Adjust for soft clipping
- Start with single-end, then expand to paired-end
- Start with known UMIs, then expand to randomers
- Millions of reads! Develop a strategy to avoid just loading everything into memory!

Your assignment is to develop a strategy to tackle this problem

- Do NOT write any code for Part 1 of this assignment!
- Define the problem, write examples
- Develop your algorithm using pseudocode
- Determine high level functions • Description
- Function header
- Test examples
- Return statement

My strategy for this assignment:

- I started by grabbing a small chunk of a .sam file from the SF-Seq assignment off of Talapas. I took 10 lines and saved them to a test file on my local computer. I'm going to manipulate this and use it as an example sam file to help me test/plan out what I'm going to do.
- I started writing pseudocode to plan how to move through the SAM file and deal with PCR duplicates

Perform this on the command line before running Deduper:

- Sort by chromosome & start position (column 3 & POS in column 4) using `samtools sort`

Run script on desired SAM file - the script will perform the following steps:

- Initialize an empty dictionary (to store read info later on)
- Initialize an empty list
- Feed in list of possible UMIs (96)
  - Fill with 96 possible UMIs

Starting with the first line:

- Store the value of the chromosome (col 3) to a variable
- Compare the UMI sequence in the header to the list of known UMIs

  - If the UMI in the header **does not** exactly match a known UMI sequence in my list:
    * *Skip* that line, continue on

  - If the UMI sequence **does** exactly match:
    * *Cut* and grab the UMI sequence (col 1), strand (col 2), chromosome (col 3), & start position (col 4)
    * Save these elements into a *tuple*

    * Add this tuple to the empty dictionary you initialized (as a key), set value to 1

    * *Write* line to output file

- If the next line contains a tuple that is **already in** your dictionary (with a value = 1):
  - *Skip* that line

- If a line contains a CIGAR string with an `S` character:
  - Isolate the number occuring before the S character (indicating how many bases were soft clipped)

  - Subtract that number from the start position in column 4
    * Save this value in the tuple

  - Compare the *new* start position (corrected for soft clipping) in the tuple to the tuples in the dictionary

  - Retain the read if it is *different* than all the other tuples encountered so far, otherwise do not retain it and move on to the next line

  - *Write* the read to an output file
- If the chromosome value on the current line **does not** equal the value stored in the chromosome variable from the previous line (indicating that the loop has moved onto a new chromosome):
  - Clear the dictionary of tuples

  - Update the chromosome variable

  - Continue