Documentation for Tfit

Michael Gohde

May 28, 2018

Abstract

Tfit is a tool that attempts to find useful parameters in sequencing datasets. This document describes the usage of Tfit in more detail than the quick start guide based on observed behavior.

Contents

1 Building and Invoking Tfit-revisions

Please see the quick start guide for documentation on building and invoking Tfit. This guide exists to detail various module behaviors discussed in the quick start guide.

2 The bidir_old module

During the course of Tfit's development history, several changes were made to how the bidir module operates. While some of these changes in behavior were desirable to the overall function of Tfit, others tended to lessen its utility in a broad context. As such, the bidir_old module was introduced to implement behavior from a known good version of Tfit such that users could choose whether they wished for Tfit to behave more like it did on publication while retaining the user interface and other backend improvements made more recently.

It appears that bidir_old, when passed the same parameters as the current bidir module, will tend to generate a substantially higher number of calls in exchange for low overall confidence. As such, it seems that this module behaves more as a filter for the remainder of Tfit. This change in behavior is explained by the following list of changes:

1. The old code sets the values of various passed parameters to those obtained from its computed average model if and only if it is passed a training file. Otherwise, it retains the values passed. This should not affect how bedgraphs are loaded later in the module, as neither the *br* nor *ns* parameters are altered.
2. The old code does not have behavior to honor the *fdr* parameter. As such, it does not compute a likelihood score distribution. It appears that the new codebase alters the behavior of its template matching algorithm based on the results of the *fdr* parameter.
3. As noted in the previous item, the old code base does not have a means of computing a likelihood score distribution. As such, instead of doing so, it *should* simply use any parameters passed to Tfit. Further analysis is necessary.

3 The bidir module

The bidir module implements bidirectional scanning from the most recent Tfit code pushed to Github. As such, it tends to perform fairly well at choosing good bidirectional reference points, though it may tend to underfit the data provided. This is likely due to two of its changes relative to older versions of the codebase:

1. The new bidir module computes a likelihood score distribution to potentially filter out "bad" calls, or those with low scores relative to the rest.
2. The new bidir module makes use of a modified backend to achieve the first point.

Unfortunately, little investigation into the exact backend changes made to achieve these changes has been conducted. As such, this is a topic of significant further review. Instead of a detailed discussion of the inner workings of the present bidir module, the workings of that which was changed relative to bidir_old will be discussed here.

3.1 Computing the average score distribution

In order to compute the likelihood score distribution, the new bidir module calls the "get_slice" function. This function performs the following:

1. Sets various internal parameters based on the values passed to Tfit. These include the "-pad", "-ns", "-sigma", "-lambda" (internally, lambda is set to the value of "-ns" divided by the value of "-lambda"), "-foot_print" (internally, fp is set to the value of "-foot_print" divided by the value of "-ns"), "-pi", "-w", and "-bct".
2. Sets up a random number generator that produces results along a uniform distribution.
3. Sets up empty lists XY and CovN. These will be used later.
4. Iterates through all the segments in the input bedgraph while altering the distribution to better fit those segments. This process follows the following steps:
    (a) Draws two random values from the uniform distribution.
    (b) Draws a segment element in proportion to the first value picked from the uniform distribution.
    (c) Scales the XN parameter of that segment by the second random value. XN should be the number of bins (defined as $(maxk - mink)/\delta$ within a given segment. This value will be used as a loop counter for the next two steps.
    (d) Sums the positive strand and negative strand data (for all?) elements in the segment.
    (e) Performs the prior step, but with a reversed second parameter. This may be because the first pass featured gaps.
    (f) Sets coverage equal to the sum of the positive and negative strand coverages obtained in the previous few steps.

(g) If the coverage meets some criteria (this is likely a filter to see whether or not this is statistically meaningful), then the BIC3 function is called. The outputs of this step are pushed into a vector.

The set of coverages found in this process is then dumped to a file ending in "_random_BIC_ratios.csv", which provides opportunities for further investigation into what is done during the average score distribution computation. Once the above process is complete, the function finds various relevant values within the BIC3 output vector. These values, in turn, form the basis for the average score distribution.

3.2 The BIC3 function

It appears that, over the course of development history, the BIC3 function has been altered on occasion. It appears to play some role in computing relevant statistics for the current bidir module. As such, it will be examined in depth here.

BIC3 appears to compute a log likelihood for a given set of statistics. After considering how BIC3 is used elsewhere in the codebase, it may be used to estimate the likelihood that a given segment or element within that segment belongs to a set of segments.

4 The model module