Annotation Agnostic Approaches to Nascent Transcription Analysis: Fast Read Stitcher and Transcription Fit

Margaret A. Gruca[1], Michael A. Gohde[2], Robin D. Dowell[1,2,3]

[1]BioFrontiers Institute; [2]Computer Science; [3]Molecular, Cellular and Developmental Biology

University of Colorado Boulder, CO USA

robin.dowell@colorado.edu

Abstract

Nascent transcription analyses provide insight into the mechanisms of gene regulation by capturing RNA transcripts pre-splicing and maturation. However, genome annotations are curated relative to mature, stable RNA transcripts and therefore annotations fail to capture the transcriptional regulatory dynamics that are observed using nascent sequencing methods. These dynamics include 5´-end initiation, RNA polymerase pausing, 3´-end transcriptional run-on, and bidirectional transcription signatures indicative of RNA polymerase loading positions. Furthermore, changes in RNA polymerase activity can be leveraged to infer participation by key transcriptional regulatory proteins. In this chapter, we describe the function of two annotation agnostic tools, Fast Read Stitcher (FStitch) and Transcription fit (Tfit), which can be used to annotate and describe genome-wide RNA polymerase transcriptional activity. These tools can be used to provide insight into transcription regulatory dynamics, as well as provide refined

annotations of both genes and putative regulatory elements for other applications such as differential transcription and motif displacement analyses.

Keywords:

Nascent transcription, GRO-seq, PRO-seq, enhancer RNAs (eRNAs), Transcription fit (Tfit), Fast Stitch Reader (FStitch), motif displacement, pausing, transcription factor, RNA polymerase, gene annotation, nuclear run-on

1 Introduction

To capture transcription dynamics across time in a broad range of cells, a number of protocols have been developed that provide information on nascent transcripts including global run-on sequencing (GRO-seq) [1], precision run-on sequencing (PRO-seq) [2], mammalian native elongating transcript sequencing (mNET-seq) [3], and chromatin run-on sequencing (ChRO-seq) [4]. These methods utilize a variety of techniques to capture transcripts as they are being produced by cellular RNA polymerases (RNAPs). Nascent transcripts are typically captured pre-splicing and maturation, and therefore are markedly distinct from mature RNA. Yet, genome annotations are curated relative to mature, processed transcripts. Consequently, annotations do not always accurately reflect the region of active transcription, as cellular polymerases may initiate upstream of the annotated 5´-end start site and often continue thousands of kilobases downstream of the annotated 3´-end of a gene. Furthermore, annotations are not sufficient to capture the large number of transcripts that occur outside of genes, particularly novel non-coding RNAs, including unannotated enhancer RNAs (eRNAs). The numbers, locations, and activity of polymerases differs between cell types, but within nascent transcription data they are readily identified via a

distinct bidirectional transcription signature. In this chapter, we describe two annotation agnostic tools that together model global transcription activity using data from nascent protocols: 1) Fast Read Stitcher (FStitch) [5,6] and 2) Transcription fit (Tfit) [7].

FStitch uses a logistic regression classifier nested within a hidden Markov model (HMM) to identify regions of active transcription. A small amount of training data is required to capture the distinct patterns of signal inherent within a particular dataset, influenced by the protocol, depth of sequencing, and complexity of the library. Once trained, FStitch can identify regions of active transcription on a per strand basis as well as regions of bidirectional transcription between the strands, many of which are eRNAs. FStitch identified active regions can be utilized downstream in a number of subsequent analysis, including as regions of interest to Tfit. Most sites of transcription are the result of the activity of RNA polymerase II (RNAPII). Tfit is a probabilistic, generative mixture model of RNAPII behavior that can annotate and characterize its activity throughout the entire genome as well as detect its behavioral changes across samples. The behavior of RNAPII is regulated at a number of steps within the well-characterized transcription cycle [8]. As several steps of the transcription cycle give rise to RNA, these steps leave distinct shapes within nascent transcription data. The Tfit model can therefore leverage these signatures within nascent data to infer sites of polymerase loading, identify regions of polymerase initiation and elongation, and quantify polymerase pausing . Furthermore, the inferred sites of RNAPII loading can be subsequently utilized to infer transcription factor activity [9].

The combination of FStitch and Tfit provide a powerful approach to understanding nascent transcription and how it changes across conditions. FStitch identifies the bounds (5´ and 3´ ends) of all transcribed regions, even when they differ significantly from annotation [10]. Notably, FStitch works on any data where the desired outcome is regions of interest, including ChIP-seq [11]. In nascent data active regions, as identified by FStitch, typically contain one or more transcripts. Tfit can dissect active regions into individual transcripts using the expected behavior of RNAPII. However, the current version of Tfit is not guaranteed to return a fit to the RNAPII

model for all regions, as some regions simply lack sufficient data for a good fit to the model or do not well reflect the anticipated shape of RNAPII data. Combinations of the two methods can be readily utilized for characterizing novel RNA transcripts, gene transcription, and identifying changes in RNAPII behavior across conditions.

2 Materials: Data and Software Requirements

Both Tfit and FStitch are stand-alone applications that take as input a coverage file (in bedGraph format) and output annotations of nascent transcription data. FStitch outputs regions of active and inactive transcription on a per strand basis whereas Tfit outputs the locations of polymerase loading as well as key characteristics (model parameters) for each unique loading position. In this chapter we discuss both the FStitch and Tfit workflows: how to go from mapped read files to model output, including a brief discussion on how to leverage those outputs in downstream analysis. For all examples, we utilize the nascent transcription data from [11], an experiment that compared Nutlin induced p53 activity to a control. The FStitch and Tfit algorithms are written in C++ and Python 3 and therefore the following compilers and compute resources are required:

2.1 Computer and Compute Resources

1. C/C++: Both packages are written in C++, so the user must have a GCC compiler >5.1 (`https://isocpp.org/`)
2. Computing Requirements: FStitch will run quickly on one core and typically requires less than 10GB of RAM. Tfit prelim will run quickly, using typically less than 30GB of RAM. Tfit model will take much longer depending on data complexity, but typically required less than 10GB of RAM. Multi-threading is available and can be user-specified for both packages.
3. Additional support code for both FStitch and Tfit is written in Python 3 (`https://www.python.org/`)

4. openmp: Tfit uses the openMPI framework to perform massive parallelization via multithreading on multiple nodes or single node systems using multiple cores (https://gcc.gnu.org/install/)
5. MPICH: Required in conjunction with openmp for parallelization, version >3.0 (https://www.mpich.org/))

2.2 Additional Software

The following is a list of software that, in addition to FStitch and Tfit, will be utilized for the full workflow:

1. BEDTools: A suite of tools used to perform coordinate math (https://bedtools.readthedocs.io)
2. samtools: Utilities for the Sequence Alignment/Map (SAM) format (http://www.htslib.org/doc/samtools.html)
3. preseq: A package used to calculate sample complexity and estimate future yields if the sample were sequenced to increased depth, available in R or for command line usage (http://smithlabresearch.org/software/preseq/)
4. BBMap Suite: A suite of various bioinformatics tools including utilities such as trimming, mapping, and post-mapping quality control (QC) (https://github.com/BioInfoTools/BBMap)
5. Integrative Genomics Browser (IGV): A genome browser application developed by the Broad Institute that includes a suite of visualization utilities. The commands listed in this manual have been tested with versions >2.3.75, and therefore may differ with earlier versions (http://software.broadinstitute.org/software/igv/)
6. Differential ATAC-seq Toolkit (DAStk): Used here as a motif displacement analysis tool (https://github.com/Dowell-Lab/DAStk)

3 Methods

3.1 Pre-Analysis: Quality Control and Data Husbandry

As both FStitch and Tfit seek to capture distinct patterns within the data, they are sensitive to the quality of the input data. Consequently, we recommend assessing the quality of the input data before utilizing either tool. In the following section, we describe our quality control analysis and provide recommendations for the quality of data needed for best results.

3.1.1 Defining model expectations by calculating sample complexity

Unfortunately, loss of data quality strongly influences the patterns observed within nascent data. For example, both distance between transcribed regions (whether or not reads overlap) and read density over regions (level of coverage) vary based on read depth (total reads sequenced) and library complexity (number of unique reads in sample). Therefore, we recommend accessing both depth and complexity prior to running either FStitch or Tfit. To this end, we recommend two tools, BBMap's pileup.sh and preseq [12], which calculate sample coverage and complexity, respectively. Both of these tools require as input a mapped read file in binary format (BAM), sorted and indexed using SAMtools [13] as follows:

```
$ samtools view -S -b -o SRR.unsorted.bam SRR.sam
$ samtools sort SRR.unsorted.bam SRR.sorted.bam
$ samtools index SRR.sorted.bam SRR.sorted.bam.bai
```

Once the BAM file has been generated, sample coverage can be assessed using BBMap's pileup.sh with the following arguments:

```
$ pileup.sh in=SRR.sorted.bam out=SRR.coverage_stats.txt
```

The output is a statistics file containing key information on the distribution and depth of reads, including average fold coverage (read density per chromosome length), chromosome length, total percent covered, percent GC of read coverage, and plus/minus read coverage. Statistics are

reported on a per chromosome basis. While all of these are valuable sample quality metrics, of particular interest for assessment prior to running FStitch is chromosome coverage. Generally, higher coverage is preferred and a minimum average coverage >3% (omitting mitochondrial reads) is recommended for running both FStitch and Tfit. Coverage can easily be visualized graphically (Figure 1A), as well as in a genome browser (Figure 2).

While coverage and complexity are interconnected in terms of quality, it is often important to assess whether further sequencing a sample with low coverage will yield more unique reads, e.g. increased sample complexity. To address this, we also recommend running preseq to calculate both current sample complexity and future yields if the sample were to be sequenced to greater depth. Preseq can be run as follows:

```
$ preseq c_curve -B -o SRR_c_curve.txt SRR.sorted.bam
$ preseq lc_extrap -B -o SRR.lc_extrap.txt SRR.sorted.bam
```

The c_curve module calculates the current sample complexity and the lc_extrap module calculates predicted future yield, if further sequencing were performed. The -B flag specifies that the input file is in BAM format. The predicted number of unique reads can be plotted against the total number of reads to visualize sample complexity (Figure 1B). It is recommended that at least 10% of 50M sequenced reads are unique for running the models. Sequencing to a greater depth such that at least 15% of 100M reads are unique in a given sample will provide the best results from both models.

As Figure 1B shows, SRR1105736 has lower complexity relative to the rest of the samples in the group which is an important consideration in modeling expectations and will be discussed in the forthcoming sections on running FStitch and Tfit.

3.1.2 Formatting the coverage file

Both FStitch and Tfit require as input a bedGraph file of the data. A bedGraph file is tab-delimited binned representation of the data, i.e. a histogram. It is therefore a more compact file format

than SAM/BAM files. Both programs specifically require a bedGraph that does not contain zeros, is a high fidelity representation of the data, and is non-normalized. While both the deepTools [14] and the BEDTools [15] suite have tools for generating a bedGraph, deepTools by default smooths the data through generation of discrete bin sizes and includes regions of zero coverage and is therefore not recommended. As such, both in terms of computational efficiency and formatting, we highly recommend using the BEDTools genomecov tool for generating bedGraphs using the following command:

```
$ bedtools genomecov -ibam SRR.bam -g SRR.bedgraph -bg -s +
```

The argument -ibam specifies the input BAM file, -g specifies the output bedGraph file, and -bg specifies the strand for the output bedGraph, in this case positive. The command should be repeated for the negative strand as follows:

```
$ bedtools genomecov -ibam SRR.bam -g SRR.neg.bedGraph -bg -s -
```

In order for FStitch and Tfit to differentiate between the strands, we negate the fourth column (coverage values) of your negative strand bedGraph. This can be done easily using awk:

```
$ awk 'BEGIN {FS = OFS = "\t"} {$4 = -$4}' SRR.neg.bedGraph >
    SRR.neg.formatted.bedGraph
```

Both FStitch and Tfit have arguments to provide separate positive and negative strand files, so at this point the necessary processing is complete for the bedGraph files. However, for visualization in Integrative Genomics Viewer (IGV) [16] we recommend that the two strand data files be concatenated into one bedGraph containing reads on both the positive and negative strands as follows:

```
$ cat SRR.pos.bedGraph < (grep -v '^@' SRR.neg.formatted.bedGraph
    | sortBed > SRR.cat.bedGraph
```

This will remove any header present in your negative strand bedGraph, concatenate positive and negative stand coverage information, and sort the bedGraph for optimal downstream

processing. This concatenated file is also accepted as input by both FStitch and Tfit in place of the individual strand files.

3.2 Using FStitch: Identifying transcribed regions

Fundamentally, FStitch is a two state Hidden Markov model that aims to distinguish between *active* and *inactive* transcription regions [6]. FStitch is comprised of three modules: 1) train, 2) segment, and 3) bidirectional (bidir). Because the characteristics of active regions are influenced by the underlying protocol, sequencing depth, and library complexity, FStitch utilizes a user defined training file to learn the key characteristics of these regions. The user defined active regions should show typical characteristics of active transcription: read dense, high-coverage contigs that span at least several hundred base pairs. Once trained, FStitch can then be utilized to segment (e.g. label) each strand's data into "*active*" and "*inactive*" regions. FStitch can then utilize the output segments called on both strands to identify regions of bidirectional transcription, defined as overlap or near overlap (default is within 300 bps) at the 5’ end of regions on opposite strands. Armed with FStitch called "active" and "bidirectional" regions, a number of downstream analysis options are subsequently available.

3.2.1 FStitch Train Module

The FStitch 'train' module requires as input two files: the data (in bedGraph format) and the training file (as BED4 for- mat). In the training file, each row is a tab-delimited instance of a single region: chromosome, start, end, and status. The status column is an indicator (0/1) as to whether the region in question is inactive or active respectively. The quality of the final FStitch output is strongly influenced by the quality and size of the training data, consequently we will describe multiple methods of creating a training file.

Producing Training Data for FStitch In this section, we will describe two ways by which you

can generate a training file that is appropriate for your data. For best results, a custom training file should be created to capture the unique characteristics of each specific dataset.

Generating good training data from scratch requires a certain degree of trial and error, however there are a few key points to keep in mind that will expedite the process. First, the BED4 format of training data does not contain strand information. Therefore, all regions within the list should originate from the same strand of data (either positive or negative). We recommend naming the training file in a manner that indicates the strand to which it corresponds. Second, regions of zero coverage are not terribly informative. Because most inactive regions will contain some noise, it is better for the training data to reflect this expectation. Therefore, we advise not to pick regions with zero coverage. Third, picking regions that are too small (rule of thumb, roughly <1 kilobase) does not accurately reflect that many transcribed regions (e.g. annotated genes) are long. Consequently, we recommend picking a mixture of regions between 1 and 200 kilobases to reflect the diversity in active transcription lengths typical in a mammalian genome. Lastly, discrete boundaries are also not necessary when calling active and inactive regions. In other words, the "start" location that is annotated as "ON" does not need to be at the precise base that signal began. With these recommendations in mind, the training data file can be generated from scratch or built upon the provided pre-configured training set. We will describe both methods and provide recommendations regarding its construction.

Pre-configured training file The simplest approach to training is to utilize the pre-configured training file provided within the train directory of the FStitch distribution. The pre-configured training file, available for the human genome version hg38 and mouse version mm10, contains annotations of twenty ubiquitously expressed genes [17] and twenty intergenic regions on the sense (pos/+) strand. In our experience, these regions provide reasonable initial training for most high quality data sets, e.g. sufficiently complex and sequenced to appropriate depth.

However, custom training data (described below) typically improves FStitch performance.

When using the pre-configured training data, it is recommended that the user first check that the default regions have adequate coverage in their specific dataset using BEDTools genomecov:

```
$ bedtools genomecov -bams [SAMPLE BAMS] -bed
    hg38_annotations.bed > sampleCoverage.bed
```

Regions of zero read coverage should be removed, but care should be taken not to remove too many regions as this dramatically reduces training effectiveness. We recommend that the training data, after customization, always contain a minimum of 15 "OFF" and 15 "ON", for a total of at least 30 regions. In our experience, the pre-configured training data is effective when all active regions have coverage and the overall read depth over active and inactive regions is over 10:1 after normalizing for bin size (*total reads/*(*end - start*)). If the pre-configured training file fails these standards, the sample should be assessed for quality (see section 3.1.1), as the data may have insufficient sample depth or complexity. Alternatively, a custom training file must be constructed.

Custom training file Creating a custom training dataset from scratch requires identifying regions of both active and inactive signal within the data. For this, we recommend utilizing the Broad's Integrative Genomics Viewer (IGV). While you can import mapped read files (typically BAM or bedGraph files) directly into IGV, numerous large files can decrease IGV's performance. As such, we recommend that the user convert the bedGraph file to TDF format, which is a binary form of the bedGraph tailored for faster access. To convert the bedGraph to a TDF, utilize IGV tools with the following command:

```
$ igvtools toTDF SRR.cat.bedGraph SRR.cat.tdf genome.chrom.sizes
```

where the file genome.chrome.sizes is a chromosome size file that corresponds to the genome to which your samples were mapped, obtained either from UCSC or provided with IGV tools.

Alternatively, you may convert the bedGraph within the IGV browser by selecting:

```
Tools → Run igvtools ...
```

from the top drop-down menu and specifying the same minimum arguments used in the command above.

Once the samples have been converted to TDF format and loaded into IGV, it is best practice to begin by looking at annotated genes that are highly expressed in the data of interest to become familiar with the typical read distribution patterns of active regions compared to inactive (or noise) regions. It is recommended that the user select between 15-20 inactive regions and 15-20 active regions.

While the user can build the required BED4 file manually, it is also possible to take advantage of IGV's built in capacity for collecting a table of regions. The coordinates for the current field of view within IGV can be added to an ongoing list using the top down menu:

```
Regions → Region Navigator ...
```

which will open a table with the necessary four columns: chromosome, start, end, and description. By clicking the "Add" button at the top, the current region shown will be added (chromosome, start and end) with the "Description" column remaining empty. In this description column, the user must add the status of the region, either a "0" or "1" for inactive (“OFF”) or active (“ON”) transcription respectively. Once multiple regions have been added to the table, the set of annotations can be exported by selecting:

```
Regions → Export Regions ...
```

and choosing where to save the training file and under what to name (must end in .bed). The file will be saved in the required BED4 format, so no further editing is required before running FStitch 'train'. Likewise, the pre-configured training regions file can be imported into IGV by going to the top drop-down menu in the program and selecting:

```
Regions → Import Regions ...
```

Regions can then be added, edited, or removed from the list to tailor the training file to your specific data using the Region Navigator, as described previously.

Training the Model Using the data bedGraph and training file, the 'train' module can be invoked using the following minimum arguments:

```
$ FStitch train --bedgraph SRR.cat.bedGraph --strand + --train
    hg38_train.pos.bed --output PROJECTNAME.hmminfo
```

where in this case the model is trained on the '+' strand of data. Alternatively, the model may be trained on the '-' strand, as is appropriate for the annotations provided by the training file. Multi-threading is also available using the -n/--threads argument, however is often not necessary as the 'train' module typically takes less than five minutes running on a single core.

The output file, which must have the .hmminfo extension for use in the FStitch 'segment' model, contains information relevant to the effectiveness of training. The header of the output file contains information pertinent to its creation including the configuration of the model, command line input, and date/time the file was generated. The first line below the header indicates whether training converged and will display 'True' if the run was successful and 'False' if it was not. While it is rare for the model not to converge, it can occur if there is not enough training data or if the input regions are inconsistent ("ON" regions resemble "OFF" regions too closely).

The other relevant value for assessing the training is in the line marked 'HMM Transition Parameters'. If your first and last values are equal to 1, this indicates that the training has converged to a single state and will fail in the segment module. If this occurs, check that the training file follows the outlined requirements or include more regions in your training.

When considering a collection of experiments, questions arise concerning how best to train and segment across the set of experiments. If the samples within the set have similar coverage and

complexity, then we encourage using the same modeling parameters across all samples, particularly when the plan is to subsequently compare transcription levels across samples. However, if the samples have very different complexities and coverage, FStitch may not be able to resolve the samples using a single trained model. In this scenario, one can train each sample individually but care must be taken to ensure that any downstream results do not arise simply from the sample specific training. Consequently, we believe best practice in this scenario is to either sequence the lower coverage sample to a greater depth (if greater complexity can be achieved), discard the sample from the analysis, or obtain a new, better quality sample.

3.2.2 FStitch Segment Module

The FStitch 'segment' module uses the output parameter file obtained from the 'train' module to annotate regions of active and inactive transcription across an entire dataset. The minimum arguments necessary to run 'segment' are as follows:

```
$ FStitch segment --bedgraph SRR.cat.bedGraph --strand (+/-)
    --params PROJECTNAME.hmminfo --output SRR.fstitch.
    {pos,neg}.bed
```

Each strand must be segmented separately, however the outputs can be concatenated such that it appears as one track in the genome browser as follows:

```
$ cat SRR.fstitch.pos.bed SRR.fstitch.neg.bed | sortBed >
    SRR.cat.fstitch.bed
```

If these results are imported into IGV, be aware that you will need to right-click on the track and select 'Expanded' to view the full annotations for both the positive and negative strands (Figure 3).

3.2.3 FStitch Bidirectional Module

The FStitch 'bidir' module uses the concatenated positive and negative strand output from the 'segment' module to annotate regions of bidirectional transcription. Bidirectional transcription is a marker of a transcription regulatory elements [18] and the RNAs arising in these regions are often referred to as enhancer RNAs (eRNAs). The output from the 'bidir' module can be used either as a pre-filter to Tfit or in differential transcription analyses, both described later in the chapter. The bidirectional module can be run using the following minimum commands:

```
$ bidir --bed SRR.cat.fstitch.bed --genes gene_ref.bed --output
    SRR.fstitch_bidirs.bed
```

The output is two files, a region file (as BED) containing the identified bidirectional regions and a statistics file (as TXT). The region file is a five-column BED file which annotates each region by chromosome, start, stop, name, and region length, respectively. This output can be uploaded into IGV for viewing (Figure 4). The stats file will also be produced including information on the parameters that were set, number of regions provided from FStitch segment output, dropped calls (typically due to length, as calls under 100bp are automatically removed), number of total called bidirectionals, and average length of the called bidirectionals.

There are a number of default settings that can also be altered to fine-tune the output relative to your data including removing regions that overlap transcription start sites (TSSs, determined using gene reference file), splitting the data into 'short' and 'long' bidirectionals, the length at which 'short' and 'long' are split, the lengths of calls which are merged, and the max length that is output. Details of all options are described in the usage statement, obtained with the option --help. For example, setting a larger 'footprint' using the -f/--footprint parameter may be useful if your data is of lower complexity as this will call bidirectionals that have a larger gap between positive and negative strand segments. Furthermore, the -p/--plot argument results in a histogram file (as HTML) of the bidirectional lengths, useful when fine tuning parameters.

3.3 Using Tfit: Inferring polymerase activity

The goal of Tfit is to apply a probabilistic mixture model of RNA polymerase II behavior to nascent transcription data in order to identify and quantify key aspects of RNAPII activity. Because Tfit seeks to capture and model a particular data distribution, data quality can significantly impact its ability to model regions of RNAPII activity.

The Tfit model describes the read distributions expected to originate from a single RNAPII loading location. Yet for an arbitrary region of data, the question arises: how many loading locations are expected? To this question Tfit utilizes a Bayesian information content (BIC) approach to determine how many loading locations best describe the region of data. Practically the process of subdividing regions is computationally expensive. Consequently, the goal is to identify regions of interest that are small (for compute efficiency) but cohesive (e.g. don't break up the signal from a single transcript). There are two primary pre-filter options available prior to running Tfit: 1) FStitch and 2) template matching (i.e. Tfit prelim module). Using FStitch as a rigorous pre-filter is the recommended approach to eliminating non-transcribed and noisy regions of the genome.

Once the regions of interest have been generated, the user can run the full Tfit model to produce a final set of annotated RNAPII model fits for each region. As nascent transcription contains transcription from all cellular polymerases but Tfit captures only RNA polymerase II behavior, Tfit is not guaranteed to return model fits for all input regions. Tfit outputs can be leveraged in downstream analyses including differential transcription analysis, changes in RNAP behavior (e.g. loading, pausing and elongation), examination of motif displacements relative to polymerase loading, and evaluation of pausing ratios.

3.3.1 Finding preliminary regions of interest For efficiency, the data should be pre-processed to identify regions of interest containing one or more RNA polymerase loading locations.

FStitch FStitch can be used as a rigorous pre-filter to identify transcribed regions within nascent transcription data. However, FStitch is sensitive to its training data and therefore is not guaranteed to provide the cohesive regions necessary for Tfit. Therefore, care must be taken to train FStitch in a manner that identifies longer contiguous regions of one or more bidirectionals. All FStitch regions should be provided as input to Tfit, including both 'short' and 'long' regions if the user specified the -split argument. We note that often transcription dense regions, such as super enhancers, cannot be parsed into individual bidirectional regions by FStitch, but can be modeled and annotated as discrete RNAPII loading events (e.g. broken into distinct sub units) using Tfit. Using the bidirectional output is relatively more conservative compared to the Tfit prelim module and therefore significantly reduces the Tfit model run-time.

Template matching An alternative pre-filter approach is known as template matching, as detailed in [7]. Encoded directly into the Tfit package, the prelim module scans the genome to identify regions that loosely match the expected model. The template matching pre-filter is included in the Tfit code base for completeness and is not necessarily a rigorous approach to identifying regions of interest. In other words, regions of low read coverage or unusual read distributions will fail to match the template and will be discarded. If a multi-threaded job is specified using the -n/-threads argument, the user must also configure the system MPI settings prior to running Tfit using the following command:

```
$ export OMP_NUM_THREADS=16
```

The minimum arguments then needed to run the prelim module are as follows:

```
$ Tfit prelim --bedgraph SRR.cat.bedGraph --jobName SRR --output
    SRR.prelim_hits.bed --logOut /log/out/dir
```

which has as input a bedGraph file (in this example called SRR.cat.bedGraph) and produces two

output files, a regions of interest (a BED4) and log file (a TXT). The regions file is a BED4 file (chr, start, stop, id) that can be subsequently used for fitting the full Tfit model. The log file includes job run information and the total number of predicted preliminary regions.

3.3.2 Tfit Model Module

The Tfit 'model' attempts to fit each region of interest to zero or more instances of the RNAPII model. The model will attempt to find the best set of parameters for μ (inferred position of polymerase loading), σ (variance in the loading position), π (strand bias), λ (length of the initiation region), and ω (pausing probability, e.g. fraction of bidirectional signal to elongation/noise signal) through maximum likelihood estimation (MLE). The model can be run using the following commands:

```
$ Tfit model --bedgraph SRR.cat.bedGraph --segment
    SRR.prelim_hits.bed --jobName SRR --output
    SRR.tfit_bidirs.bed --logOut /log/out/dir
```

which has as input both the nascent data (as bedGraph), the regions of interest (--segment) and produces as output a BED file containing annotated regions of the model fit (SRR.tfit_bidirs.bed). Additionally, the full model estimates are written to a log file (/log/out/dir/jobName_models_MLE.tsv) which gives a detailed account for each region of interest of the parameters and log-likelihood score. Full details of the log file contents are included in the Tfit GitHub documentation.

The main output file, SRR.tfit_bidirs.bed (a BED file) corresponds to individual regions that fit the underlying polymerase model and can be readily viewed in IGV (Figure 5). Within the BED file, the first three columns are the standard BED required chromosome, start and stop for each region. The center of each interval is the RNA polymerase loading position (μ) and the width corresponds to the initiation region (i.e. 2 * σ + λ). The fourth column provides a label for each output bidirectional. Each label reflects a standard format that links the outputs to each input

region (see the Tfit GitHub documentation for more details). Columns 5 through 10 are μ (polymerase loading), σ (standard deviation from μ), ω (pausing probability), λ (exponential decay rate, e.g. length of initiation), π (strand bias), positive strand coverage, and negative strand coverage, respectively.

As with the prelim module, the model module can be multi-threaded using the -n/--threads argument as long as the MPI setting are specified for the system. Additionally, a file containing annotated bidirectionals can optionally be provided to refine Tfit's modeling parameters to your dataset and regions of interest. This may be an NCBI RefSeq file containing transcription start sites or hand-annotated bidirectionals in BED3 format (chr, start, stop) acquired using a genome browser such as Integrative Genomics Viewer (IGV). For detailed instructions on annotating and exporting regions of interest using IGV, see section 3.2.1.

3.4 Downstream Applications

The goal of FStitch is the identification of regions of active transcription whereas Tfit attempts to annotate RNA polymerase activity. While the outputs of both software can be analyzed in any number of ways, here we highlight a few of the most common downstream applications.

3.4.1 Understanding changes in RNA polymerase II activity

The outputs of Tfit provide a quantification on RNA polymerase II behavior within a particular dataset. RNA polymerase II loads at a number of locations genome wide and cycles through three distinct phases: initiation, elongation, and termination [19]. Each Tfit output is a single fit of the model and has associated a number of model parameters (see Figure 8A-E). The regions output by Tfit correspond to the inferred initiation region ($2*\sigma + \lambda$) and have at their center the inferred location of RNA polymerase II loading (μ). The loading position (μ) can be utilized in a number of subsequent analysis including motif displacement analysis (see Section 3.4.3, Figure 8F and G), comparison to annotation to assess 5´ end usage, or differential transcription (see

Section 3.4.2).

The pausing probability (ω) calculated by Tfit (Figure 8D) refers to the fraction of RNA polymerase that pause and is somewhat analogous to the more commonly used pausing ratio. The distinction is that the pausing probability (ω) accounts for the inherent overlap of initiation and elongation regions whereas pausing ratios are typically calculated from discrete windows. When assessing pausing, one can either evaluate the pausing probability directly or utilize the RNA polymerase II loading position (μ) as a reference point for construction of more traditional pausing ratio windows [20].

As Tfit does not currently contain a model of termination, we recommend utilizing FStitch to identify the 3´ end of transcribed genes. By intersecting FStitch called regions of transcription with annotations, the extent to which an elongation region extends beyond the annotated cleavage site can be readily determined.

3.4.2 Differential Transcription Analysis

There have been numerous methods developed for assessing differential expression of mature RNA from read count data [21, 22]. However, nascent transcription has unique properties relative to steady state mature messenger RNA. Notably, nascent transcripts are pre-splicing, have distinct RNA polymerase initiation peaks, and terminate far beyond the canonical cleavage site [6].

When assessing gene level differential transcription from nascent data, one must first decide whether to assess patterns of change for initiation regions, elongation regions, annotated genes, or the full region of transcription (initiation + elongation), which often extends well beyond the annotated cleavage site. As described above, a combination of FStitch and Tfit can be utilized to delineate these distinct regions. Given the desired regions, read counts can be gathered and differential transcription assessed using the program of your choice such as DESeq/DESeq2 or edgeR [23-25].

Most sites of RNA polymerase II loading, however, do not give rise to stable messenger RNAs

[7]. Instead, many sites of bidirectional transcription correspond to transcribed regulatory elements including enhancers. There are multiple ways to identify these regions from nascent data. Most regions of bidirectional transcription are captured with the FStitch bidir module. However, the regions output from FStitch may overlap, will not necessarily be identical across samples, and longer regions may include multiple bidirectionals or a significant portion of elongating transcription. In these long regions, Tfit can be utilized to identify discrete bidirectionals. It is worth noting that Tfit called bidirectionals are, in general, shorter than those identified by FStitch (see example in Figure 5).

If the interest is only in changes at non-gene associated regulatory regions, we recommend only using 'short' bidirectional calls from FStitch and removing overlap with annotated transcription start sites (TSS). This will minimize the variance in region sizes as well as reduce the total number of regions considered. However, it is important to note that because these regions are small, there will likely be a large degree of variance in read count over bidirectionals. Consequently, it is highly encouraged to have multiple replicates to do these types of analyses. When dealing with replicates, we recommend that the user concatenate and merge the bidirectional output BED files from all samples into one file in order to ensure that multiple regions are not being counted twice and to capture all potential bidirectional regions in the dataset (see Figure 7).

3.4.3 Inferring Transcription Factor Activity Through Motif Displacement Analysis

Regulatory regions, including enhancers, are dense with transcription factor (TF) binding sites. Activity of a bound transcription factor results in changes to nearby transcription. Motif displacement (MD) analysis uses the Tfit identified sites of RNA polymerase II loading (μ) and a collection of TF motifs [26] to infer when a transcription factor's activity is altered by the perturbation [9]. The MD score is the ratio of TF motifs located within a 150bp window around μ relative to the larger local background (a 1500bp window). In the original description, MD scores were compared between conditions (e.g. control versus treatment).

Typically, however, we are interested in identifying which transcription factors are driving the patterns of differential transcription observed. Therefore, we now recommend a modified approach using DAStk [27] to assess MD scores across conditions. While DAStk was originally developed for ATAC-seq data analysis, it can assess MD scores relative to any feature of interest, e.g. μ for nascent transcription or the peak summit for ChIP-seq. Using DAStk we compare MD scores between the set of differential transcribed bidirectionals and the background set of unchanged bidirectionals. In other words, first determine the differential transcription signal at all bidirectionals (see Section 3.4.2) and sort regions by p-value. Using a loose cutoff on p-value (in Figure 8F and G we used p-value < 0.2), we then compare the MD score for the differentially transcribed set of bidirectionals to the MD score of the remaining (unchanged) bidirectionals. The relatively loose p-value cutoff ensures that all bidirectionals with alternations in transcription, even modest changes, are appropriately counted towards changes in transcription factor activity. Overall, this modified approach to MD score analysis better accounts for replicates and the inherent position specific GC bias (Figure 8H) observed at bidirectionals. In our experience, this approach accurately identifies the transcription factors changing between two conditions, often with better signal to noise characteristics than our original formulation [9].

3.5 Availability

Both FStitch and Tfit are publicly available and open source. Additional documentation as well as installation guides are available on the main GitHub repository pages (`https://github.com/Dowell-Lab/FStitch` & `https://github.com/Dowell-Lab/Tfit`). Both programs can be cloned directly from these sources and compiled using a GCC compiler version >5.1 and executing the shell script 'setup.sh'. Tfit also requires

MPICH to be compiled and must be set in your PATH.

Alternatively, FStitch and Tfit are available as Docker containers that can be pulled from the Docker image repository (`biofrontiers/fstitch_tfit`) or built following the instructions detailed on GitHub. An Amazon machine instance (AMI) is also available to run both FStitch and Tfit, with detailed instructions on GitHub.

Finally, FStitch, Tfit, and DAStk are also integrated into our full nascent data processing Nextflow pipeline, Nascent-Flow (`https://github.com/Dowell-Lab/Nascent-Flow`; `doi:10.17605/OSF.IO/NDHJ2`). It is important to note that the software packages will run with default settings within the Nextflow pipeline and it is therefore recommended that the user refine these settings as needed for experiment specific data analysis.

Acknowledgments

This manuscript will be included in an upcoming volume of Methods in Molecular Biology (MiMB) on Nascent RNA.

References

1. L. J. Core, J. J. Waterfall, and J. T. Lis, “Nascent RNA sequencing reveals widespread pausing and divergent initiation at human promoters,” Science, vol. 322, pp. 1845–1848, December 2008.
2. H. Kwak, N. J. Fuda, L. J. Core, and J. T. Lis, “Precise maps of RNA polymerase reveal how promoters direct initiation and pausing,” Science, vol. 339, no. 6122, pp. 950–953, 2013.3. T. Nojima, T. Gomes, A. R. F. Grosso, H. Kimura, M. J. Dye, S. Dhir, M. Carmo-Fonseca, and N. J. Proudfoot, “Mammalian NET-seq reveals genome-wide nascent transcription coupled to RNA processing,” Cell, vol. 161, no. 3, pp. 526–540, 2015.
3. T. Nojima, T. Gomes, A. R. F. Grosso, H. Kimura, M. J. Dye, S. Dhir, M. Carmo-Fonseca, and

N. J. Proudfoot, "Mammalian NET-seq reveals genome-wide nascent transcription coupled to RNA processing," Cell, vol. 161, no. 3, pp. 526-540, 2015.

4. T. Chu, E. J. Rice, G. T. Booth, H. H. Salamanca, Z. Wang, L. J. Core, S. L. Longo, R. J. Corona, L. S. Chin, J. T. Lis, H. Kwak, and C. Danko, “Chromatin run-on reveals nascent RNAs that differentiate normal and malignant brain tissue,” Nature Genetics, 50, pp. 1553–1564, Oct 2018.
5. J. Azofeifa, M. A. Allen, M. Lladser, and R. Dowell, “Fstitch: A fast and simple algorithm for detecting nascent RNA transcripts,” in Proceedings of the 5th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics, BCB ’14, (New York, NY, USA), pp. 174–183, ACM, 2014.
6. J. G. Azofeifa, M. A. Allen, M. E. Lladser, and R. D. Dowell, “An annotation agnostic algorithm for detecting nascent RNA transcripts in GRO-seq,” IEEE/ACM Transactions on Computational Biology and Bioinformatics, vol. 14, pp. 1070– 1081, Sep 2017.
7. J. G. Azofeifa and R. D. Dowell, “A generative model for the behavior of RNA polymerase,” Bioinformatics, vol. 33, pp. 227–234, Sep 2016.
8. N. J. Fuda, M. B. Ardehali, and J. T. Lis, “Defining mechanisms that regulate RNA polymerase II transcription in vivo,” Nature, vol. 461, pp. 186–192, Sep 2009.
9. J. G. Azofeifa, M. A. Allen, J. R. Hendrix, T. Read, J. D. Rubin, and R. D. Dowell, “Enhancer RNA profiling predicts transcription factor activity.,” Genome research, Feb 2018.
10. G. J. Sanchez, P. A. Richmond, E. N. Bunker, S. S. Karman, J. Azofeifa, A. T. Garnett, Q. Xu, G. E. Wheeler, C. M. Toomey, Q. Zhang, R. D. Dowell, and X. Liu, “Genome-wide dose-dependent inhibition of histone deacetylases studies reveal their roles in enhancer remodeling and suppression of oncogenic super-enhancers,” Nucleic Acids Research, vol. 46, pp. 1756–1776, Dec 2017.
11. M. A. Allen, H. Mellert, V. Dengler, Z. Andryzik, A. Guarnieri, J. A. Freeman, X. Luo, W. L. Kraus, R. D. Dowell, and J. M. Espinosa, “Global analysis of p53-regulated transcription

identifies its direct targets and unexpected regulatory mechanisms," eLife, vol. 3, p. e02200, 2014. doi: 10.7554/eLife.02200.

12. T. Daley and A. D. Smith, "Predicting the molecular complexity of sequencing libraries," Nature Methods, vol. 10, p. 325, Feb 2013.
13. Li H., Handsaker B., Wysoker A., Fennell T., Ruan J., Homer N., Marth G., Abecasis G., Durbin R. and 1000 Genome Project Data Processing Subgroup (2009) The Sequence alignment/map (SAM) format and SAMtools. Bioinformatics, 25, 2078-9. Jun 2009.
14. F. Ramírez, S. Diehl, T. Manke, F. Dündar, and B. A. Grüning, "deepTools: a flexible platform for exploring deep-sequencing data," Nucleic Acids Research, vol. 42, pp. W187–W191, 05 2014.
15. A. R. Quinlan and I. M. Hall, "BEDTools: a flexible suite of utilities for comparing genomic features," Bioinformatics, vol. 26, pp. 841–842, 01 2010.
16. J. T. Robinson, H. Thorvaldsdóttir, W. Winckler, M. Guttman, E. S. Lander, G. Getz, and J. P. Mesirov, "Integrative genomics viewer," Nature Biotechnology, vol. 29, p. 24, Jan 2011.
17. D. Ramsköld, E. T. Wang, C. B. Burge, and R. Sandberg, "An abundance of ubiquitously expressed genes revealed by tissue transcriptome sequence data," PLOS Computational Biology, vol. 5, pp. 1–11, 12 2009.
18. C. G. Danko, S. L. Hyland, L. J. Core, A. L. Martins, C. T. Waters, H. W. Lee, V. G. Cheung, W. L. Kraus, J. T. Lis, and A. Siepel, "Identification of active transcriptional regulatory elements from GRO-seq data," Nat Meth, vol. 12, pp. 433– 438, 05 2015.19.
19. J. Shandilya and S. G. E. Roberts, "The transcription cycle in eukaryotes: From productive initiation to RNA polymerase II recycling," Biochimica et Biophysica Acta (BBA) - Gene Regulatory Mechanisms, vol. 1819, no. 5, pp. 391–400, 2012.
20. K. Adelman and J. T. Lis, "Promoter-proximal pausing of RNA polymerase II: emerging roles in metazoans," Nat Rev Genet, vol. 13, pp. 720–731, 10 2012.
21. J. Costa-Silva, D. Domingues, and F. M. Lopes, "RNA-Seq differential expression analysis: An extended review and a software tool," PLOS ONE, vol. 12, no. 12, pp. 1–18, 2017.

22. C. Soneson and M. Delorenzi, "A comparison of methods for differential expression analysis of RNA-seq data," BMC Bioinformatics, vol. 14, p. 91, Mar 2013.

23. S. Anders and W. Huber, "Differential expression analysis for sequence count data," Genome Biology, vol. 11, no. 10, p. R106, 2010.

24. M. I. Love, W. Huber, and S. Anders, "Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2," Genome biology, vol. 15, no. 12, p. 550, 2014.

25. M. D. Robinson, D. J. McCarthy, and G. K. Smyth, "edgeR: a Bioconductor pack-age for differential expression analysis of digital gene expression data," Bioinformatics (Oxford, England), vol. 26, pp. 139–140, Jan 2010.

26. S. A. Lambert, A. Jolma, L. F. Campitelli, P. K. Das, Y. Yin, M. Albu, X. Chen, J. Taipale, T. R. Hughes, and M. T. Weirauch, "The human transcription factors," Cell, vol. 172, no. 4, pp. 650–665, 2018.

27. I. J. Tripodi, M. A. Allen, and R. D. Dowell, "Detecting differential transcription factor activity from ATAC-Seq data.," Molecules (Basel, Switzerland), vol. 23, May 2018.

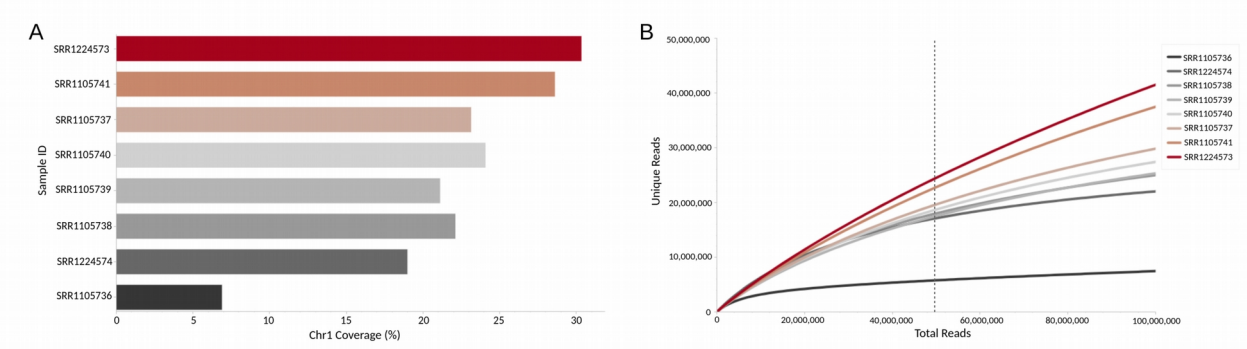

Fig. 1. Plotting Coverage and Complexity (A) Fraction of chromosome (as percentage) covered (for Chromosome 1) in multiple samples [10], generated using data from BBMap's pileup.sh. (B) Comparative sample complexity curves generated from the preseq lc_extrap module output where the y-axis represents the predicted number of unique reads at any corresponding sampling depth (total reads) on x-axis. We recommend a 50M read sampling depth with at least 10% unique reads (dashed line).

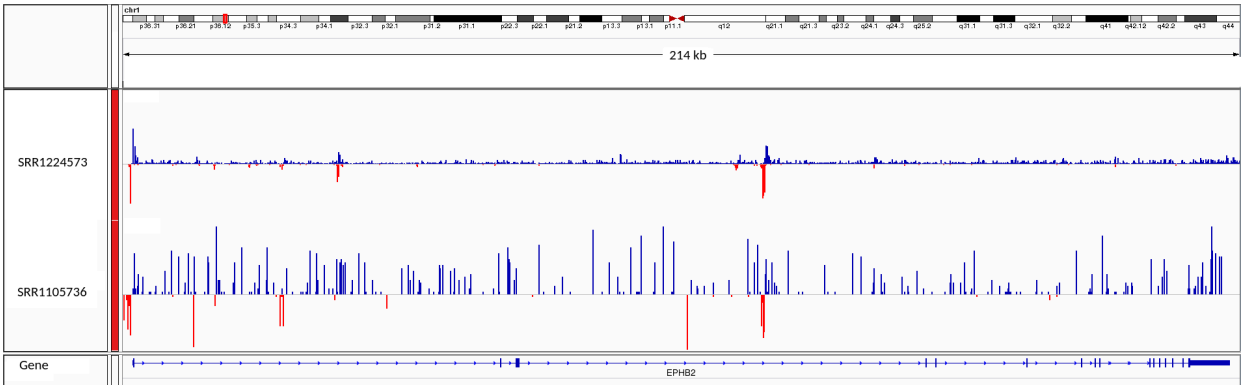

Fig. 2. Visualizing Sample Complexity Visual example of the differences in coverage and complexity between two samples within the same experiment over the gene EPHB2 using IGV. Samples SRR1105736 and SRR1224573 have 6.4% and 30.9% cover- age over chromosome 1 and 5.8M and 24.5M unique reads per 50M reads, respectively. SRR1105736 therefore has lower coverage and complexity relative to the other samples in this experiment (hg38; group auto-

scaled; chr1:22,708,838-22,923,500; y-axis [-0.405 - 0.49]).

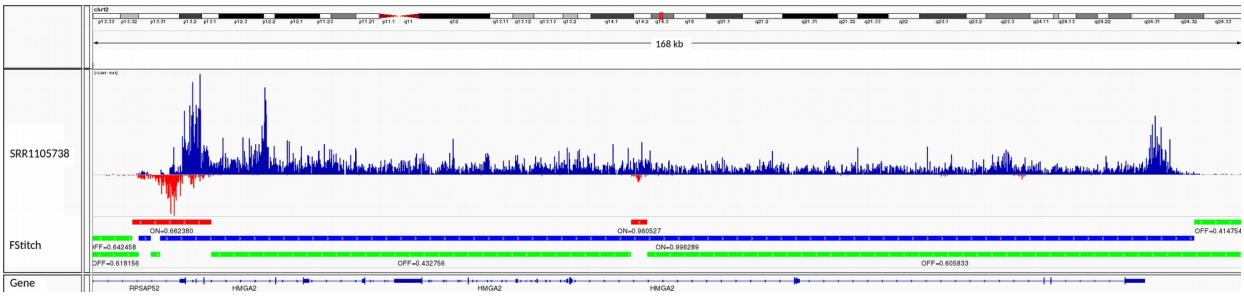

Fig. 3. FStitch segment output FStitch segment module output displayed with concatenated positive/negative strand. Generated using the pre-configured hg38 training file included with the FStitch module. The green regions displayed on the FStitch track ("expanded" option selected in IGV) represent the strand-specific transcription- ally inactive ("OFF") calls whereas the red and blue regions represent transcription- ally active "ON") regions for the negative and positive strands, respectively (hg38; chr12:65,813,792-65,982,488; y-axis [-2.243 - 5.41]).

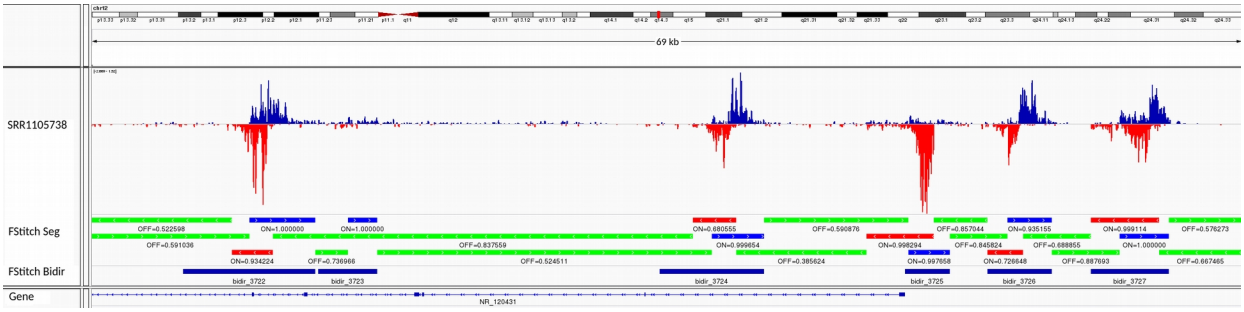

Fig. 4. FStitch bidir module output FStitch bidirectional annotation of a super- enhancer region visualized in IGV (hg38, chr12:65,599,070-65,669,060, y-axis [-2.669 – 1.52]).

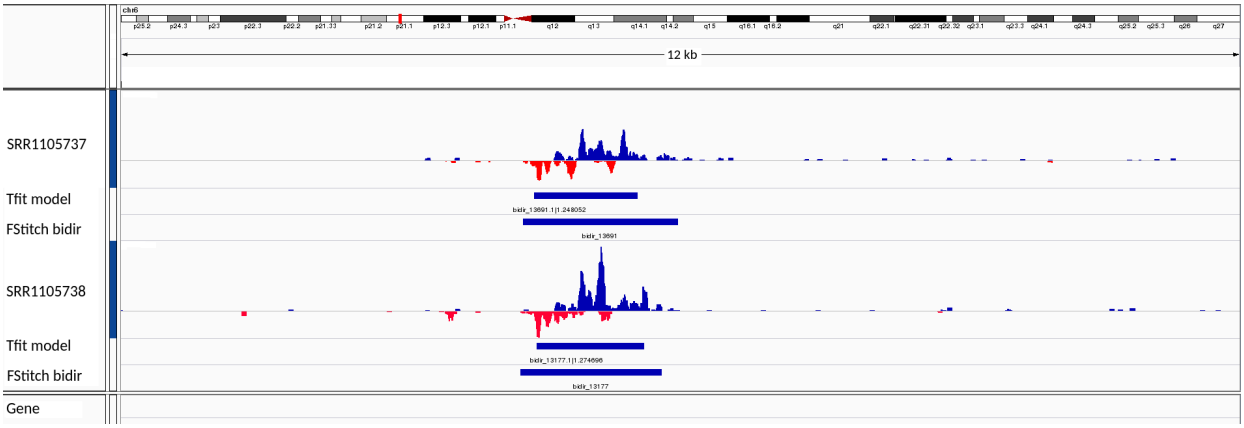

Fig. 5. Tfit model output Tfit model BED file output generated using FStitch bidir as the preliminary filter, loaded into IGV for visualization. Tfit output labels have two parts: the preliminary region modeled | BIC score for the region (hg38; chr6:42,542,533- 42,555,332; y-axis [-0.351 - 0.82]).

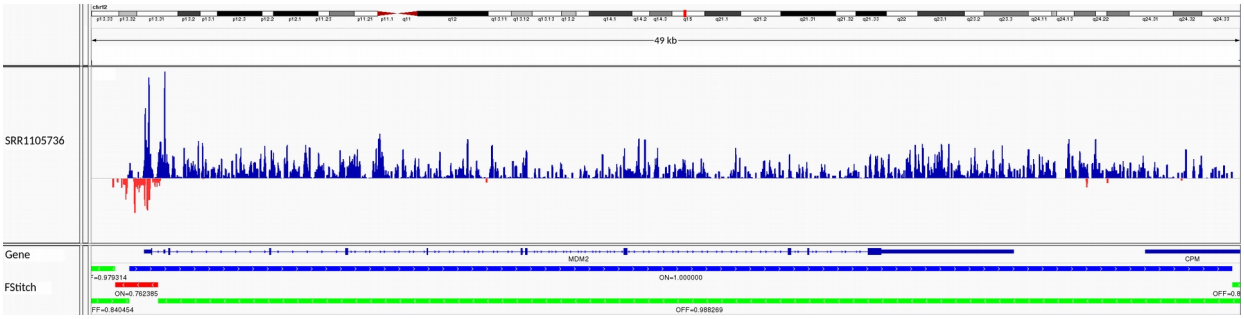

Fig. 6. Capturing transcriptional annotation over a gene region FStitch segment can be used to capture full gene-level transcription annotation for application in differential transcription analysis. Note that polymerase transcription run-on continues 10 kb past the annotated 3 end of MDM2 (hg38; chr12:68,806,508-68,855,728; y-axis [-1.172 - 3.58]).

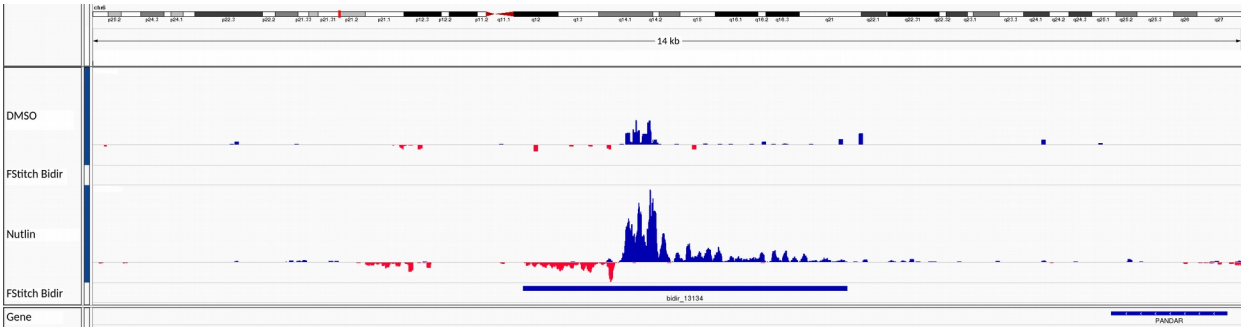

Fig. 7. Using bidirectional annotations to capture differential transcription in regulatory elements Due to differing levels of transcription between DMSO (top = SRR1105736) and Nutlin-3a treated samples (bottom = SRR105738), the bidirectional region is only captured by the FStitch model in one sample. We therefore recommend the user merge annotations within an experiment to both be able to capture highly differentially transcribed bidirectionals and prevent counting the same region twice (hg38; group-autoscaled; chr6:36,662,485-36,675,284; y-axis [-0.797 - 2.82]).

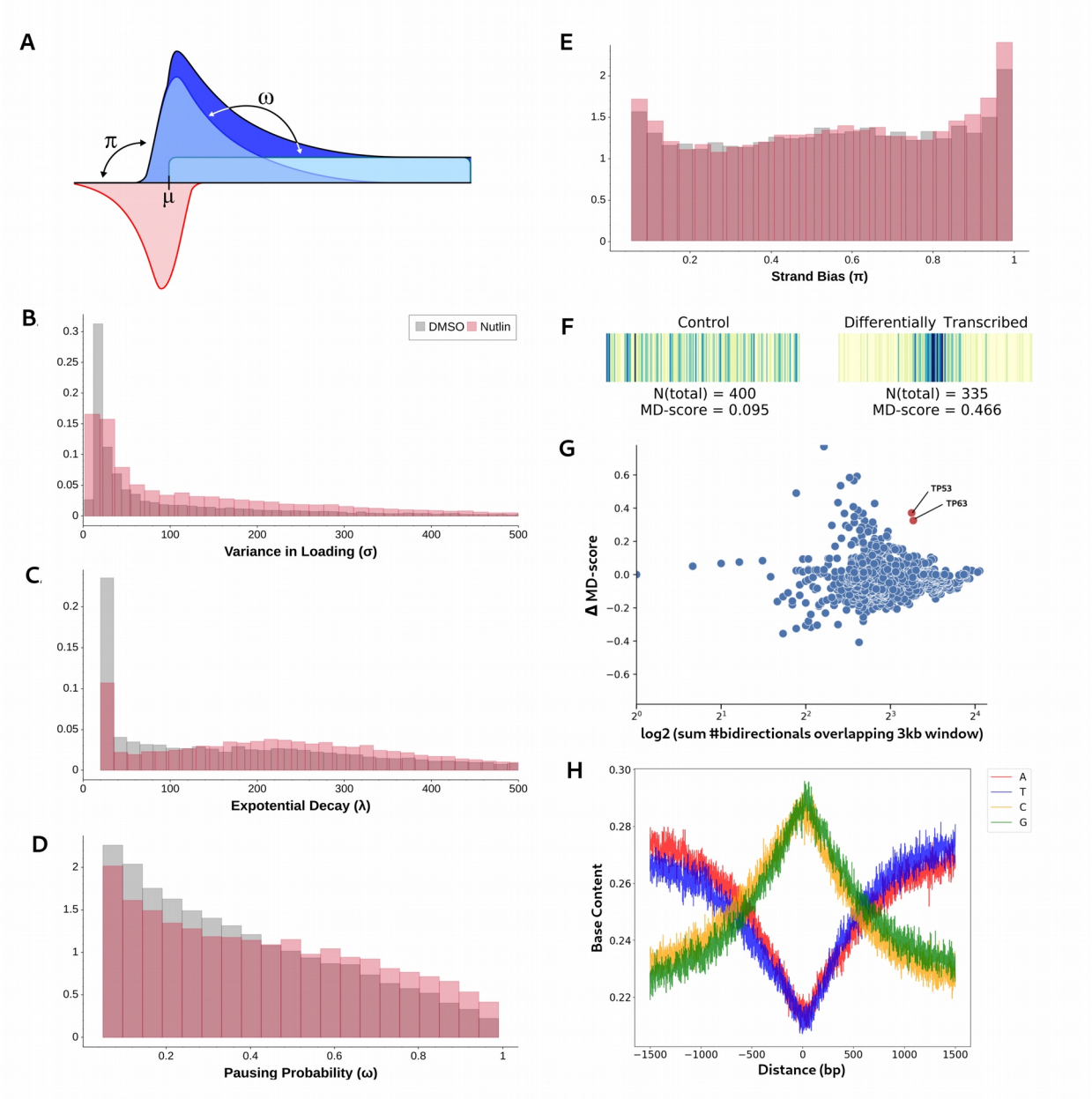

Fig. 8. Downstream analysis using Tfit model output (A) Cartoon representation of the Tfit model parameters highlighting the position of polymerase loading (μ), the strand bias (π) and pausing probability (ω) parameters. (B-E) Histograms representing variance in RNAPII loading position (B, σ) exponential decay (C, λ), pausing probability (D, ω), strand bias (E, π) between DMSO (grey) and Nutlin (red) treated samples for all bidirectionals (F) Motif displacement plots (as

heatmaps) for the TP53 motif (ID: M06704_1; increasingly dark blue indicates more instances of the motif), comparing control (left: unchanged in transcription) to differentially transcribed (right). Heatmaps are centered on μ showing a window of 3 kb. (G) Change in MD score plot using motifs from a curated database (http://humantfs.ccbr.utoronto.ca/cite.php, v1.01). Motif hits generated using FIMO with a p-value threshold of 1e-6. MD scores with a positive differential MD score (enrichment) and p-value < 1e-200 are highlighted in red. (H) Average fractional base content per position over a 3000 bp window centered at μ of all Tfit calls in the dataset, demonstrating a strong GC bias.