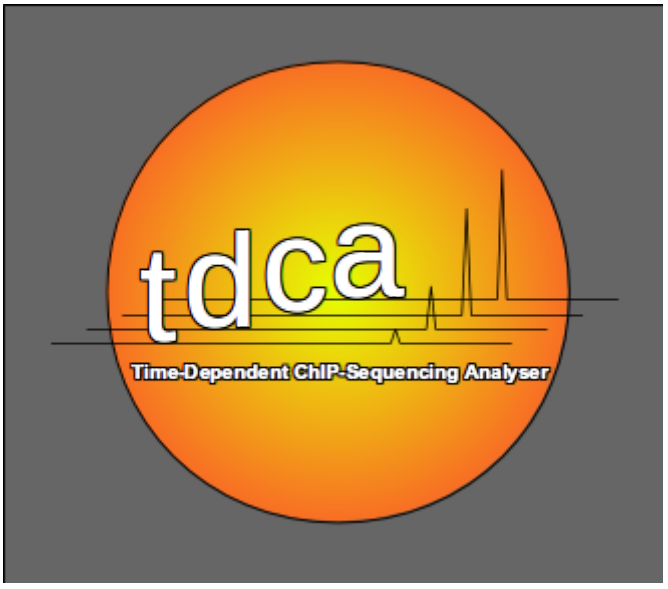
Time-Dependent ChIP-Sequencing Analyser (TDCA)
Manual for Version 1.1.0



Authors: Mike Myschyshyn, Marco Farren-Dai, Tien-Jui Chuang and David Vocadlo.
Download: https://github.com/TimeDependentChipSeqAnalyser/TDCA
Contact: mmyschyshyn@gmail.com; dvocadlo@sfu.ca

Index

1. Overview

1.1 Background

Chromatin immunoprecipitation followed by sequencing (ChIP-seq) is an established and robust method to generate genome wide maps of DNA binding proteins. Recently, new methods have been developed allowing time resolved ChIP-seq and ChIP-seq-like experiments to be conducted, effectively allowing protein-DNA binding dynamics and other DNA interaction dynamics to be established. As a response to the increasing potential of time course ChIP-seq experiments, we developed a robust software specializing in time course ChIP-seq, as well as other time resolved sequencing analysis. Our software, Time-Dependent ChIP-seq Analyser (TDCA), produces biologically relevant output informing users of protein-DNA binding dynamics. TDCA reads alignment data in BAM file format and genomic coordinates in BED file format.

1.3 Implementation

TDCA functionalities were developed using C++ and its graph features were built-up under R. TDCA uses samtools for BAM file coverage calculations and bedtools for peak intersection with genome features. TDCA makes various calls to the command line while running such as sed, awk, find, and others. Some hard coded files are created as well which users should keep in mind while using TDCA in pipelines. Samtools, bedtools, and R must be accessible from the command line.

1.4 Note about the manual

The proceeding contents of the manual contain example commands including TDCA usage. If a line starts with the $ character, it is meant to imply a command run in the terminal with the appropriate files available in the working directory. The following instructions assume a basic understanding of terminal navigation and command execution. This manual is intended to teach uses how to effectively use TDCA.

2. Installation

2.1 Software Requirements

Before installing TDCA, we require users to install the dependencies listed below, in Table 1:

Table 1. TDCA dependencies and download links

Name	Download Link
R	https://cran.r-project.org/bin/windows/base/ (Windows)
R Package plot3D	install.packages(“plot3D”)
R Package drc	install.packages(“drc”)
R Package rgl	install.packages(“rgl”)
R Package ggplot2	install.packages(“ggplot2”)
R Package scales	install.packages("scales")
R Package grid	install.packages("grid")
R Package reshape	install.packages("reshape")
R Package scatterplot3d	install.packages("scatterplot3d")
Bedtools	https://github.com/pezmaster31/bamtools
Samtools	https://github.com/samtools/samtools

TDCA requires bedtools (Quinlan and Hall, 2010) and samtools (Li *et al*., 2009) be installed on the user's local computer and set on the environmental variables. TDCA calls these programs in many of its calculations. The user can check if bedtools and samtools is accessible globally by typing “samtools” and “bedtools” in the command line. If the programs are accessible, relevant information regarding the program will print. Alternatively, the user can check their bashrc file, or equivalent, with the following command:

```
$gedit ~/.bashrc
```

An alternative text editor to gedit may be used. If samtools and bedtools are set in environmental variables then the path to each of these program directories will be documented on a line in the bashrc file like this:

```
export PATH=$PATH:/software/folder/bedtools/bin
export PATH=$PATH:/software/folder/samtools/bin
```

Although depending on how the softwares were installed, the above lines may not be present in your bashrc file. Again, the simple way to check if these programs are installed and accessible to TDCA is by typing “samtools” and bedtools” in the command line, because this is how TDCA calls these programs.

TDCA uses the R packages drc (Ritz,C. *et al*. 2015) for sigmoidal curve fitting and ggplot2 for graphical output (H. Wickham, 2009). Installation of R packages can be conducted as follows:

$R
Installing package:
>install.packages(“package_name”)
Checking if package is installed:
>library (“package_name”)

To our knowledge, the R package drc which is used for curve fitting requires R version 3.3.1 or later.

2.1 TDCA Installation Guide on Linux

Once all the dependencies are installed users can proceed to TDCA installation. Users can download a tar file from:
https://github.com/TimeDependentChipSeqAnalyser/TDCA

Unpack and navigate to the tdca directory. Assuming the unpacked tdca folder is in the home directory:
$cd home/tdca
Run make:
$make
Now add tdca directory to environmental variables to allow accession from any directory:
$gedit ~/.bashrc
Write this line: export PATH=$PATH:home/tdca

Once tdca is added to the environmental variables, the program can be used from any directory like so:
$tdca <options>

If tdca is not added to the environmental variables, the full program path must be specified from the working directory. Ex:
$/home/tdca/tdca <options>

2.2 TDCA Installation Guide on Windows

2.2.1 Virtual Box

1. Download Oracle VM VirtualBox

2. Download Ubuntu Desktop
3. Install VirtualBox
4. In VirtualBox Manager, click New
5. Give a name to the operating system and select Linux for Type and Ubuntu 32/64 bits depending on the Windows OS
6. For RAM Memory size, give above 4GB (4096MB)
7. For Hard Disk, select Create a virtual hard disk now
8. For Hard Disk File Type, select VDI
9. Select Dynamically allocated
10. For File Location and Storage, give above 32 GB
11. In Storage, click on [Optical Drive] and select Choose a disk image
12. Open the ubuntu-version-desktop-amd32/64.iso
13. Click Start to initiate Ubuntu system in VirtualBox
14. Follow the steps to complete the Ubuntu installation
15. Once Ubuntu is successfully installed, run the command prompt and install all the necessary softwares for TDCA. Please read the installation guide on Linux in 2.1

2.3 Using TDCA

Calling tdca without any arguments results in an output of the flag options and a brief description, as shown in Table 2.

Table 2. TDCA options and brief description

Command	**Description**
-v	Display program version and exit program.
-h	Display a detailed list of all command line options and exits the program.
-bam	User specified folder containing sorted BAM turnover files including index files.
-bed	User specified BED file containing loci of interest.
-i	User specified folder containing sorted input BAM turnover files including index files.
-g	Genome name. Currently supported: mm10, mm9, hg38, hg19, dm6, dm3, ce11.
-3d	User specified gene file containing RefSeq gene names.
-s	Plateau range threshold (allowable range from 0.5-0.95). Default = 0.85.

-t	Leading/trailing threshold consideration for data modelling (allowable range from 0-2). Default = 1.
-name	User specified name for output files. Default = turnover.exp.
-model	Data modelled based on prediction. Default is no data modelling.
-poisson	Data modelled to 3 parameter sigmoidal curve assuming Poisson distributed coverage.
-dm	Data matrix used to normalize user defined input files.
-prenorm	User defined pre-normalized read counts with genome coordinates.
-nonorm	Read coverage will not be normalized based on sequencing coverage of non-peak loci.
-L5	Model data to five parameter sigmoids instead of the default four parameter sigmoids.
-proc	Explicitly state number of processors to use.
-lin	Perform linear regression on time course data.

3. Core Algorithm Description

TDCA reads genomic coordinates provided in BED file format, extracts the sequencing coverage at coordinates from the provided time course BAM formatted sequence files with Samtools, and models the sequencing coverage to four (default) or five parameter (4P or 5P) sigmoidal curves using the drc R package. The equation and description of parameters for a 5P sigmoid are shown in equation 1. Figure 1 shows what sigmoidal curves look like when the incorporation rate index (IRI) parameter is altered. Figure 2 shows the effect of changes in the asymmetry value *f*.

$$y = d + \frac{a-d}{\left(1+e^{b(x-c)}\right)^{f}} \qquad \text{Eqn. 1}$$

Where,
a = Lower asymptote
b = Incorporation rate index (IRI, a measure of the slope at the inflection point)
c= Inflection point when f=1 (also the time at which the curve reaches th when f=1)
d = Upper asymptote
f = Asymmetry factor

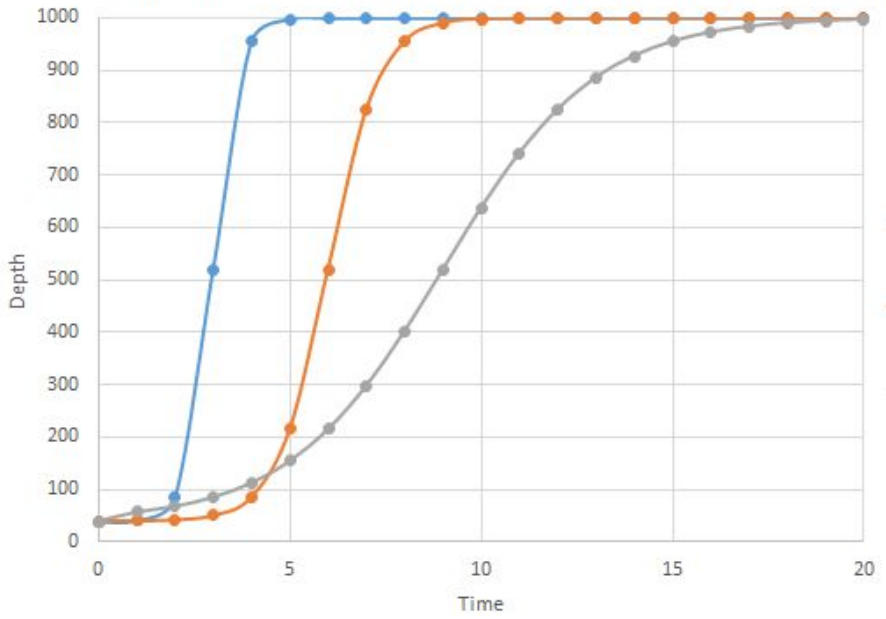


Figure 1. Graphical representation of the effect of changes in IRI (Hill coefficients) on the shape of 5P curves. For each of the three curves, the lower asymptote is held at 40, the upper asymptote at 1000, and the asymmetry factor at 1. The blue curve has a IRI of 3 and inflection point of 3. The orange curve has a IRI of 1.5 and inflection point of 9. The grey curve has a IRI of 0.5 and inflection point of 15.

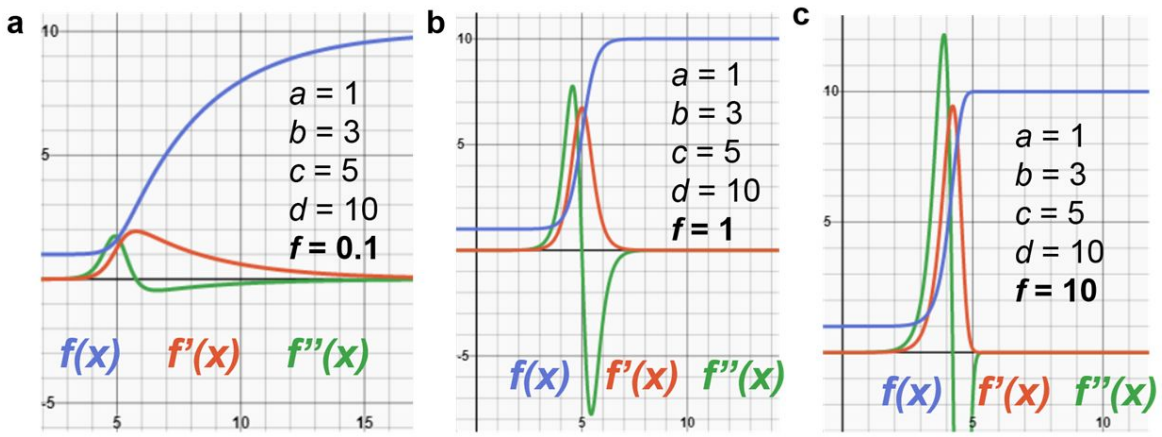


Figure 2. Graphical representation of the effect of changes in asymmetry value f on *y* in blue as well as the first derivative *y'* in red and second derivative *y''* in green. All other values for y remain the same as indicated. (**A**) In cases where $0 < f < 1$, the inflection point (given by the maximum in *y'* and the root of *y''* occurs closer to the lower asymptote. (**B**) For $f = 1$: y is rotationally symmetric about the inflection point. The increase in rate is the same as decrease in rate (as seen by the vertical symmetry of *y'*) such that inflection point occurs exactly in between the lower and upper asymptote. (**C**) For $f > 1$: *y* inflection point occurs closer to the upper asymptote). Plots were generated using Derivative Calculator at https://www.derivative-calculator.net/

In the case of a positive *b* value as in the Figure 2 plots, $0 < f < 1$ results in an earlier inflection point which could be interpreted biologically as a fast induction mechanism that reaches its maximal rate quickly but takes a relatively long time to saturate. In fact, the behaviour of this curve is very similar to that of an inverse negative exponential function which has been used previously to fit chIP-Seq data (Deal, 2010). Conversely for $f > 1$, this results in a later inflection point, which could be interpreted as a slower induction mechanism that reaches its maximal rate slowly but saturates quickly once it reaches the inflection point. Note that the recommended default setting for f is fixed 1 and changing this setting should only be done by expert users with a clear biological rationale since a variable f may lead to misleading fitting.

During fitting, each locus in a time course ChIP-seq experiment is reduced to one of six characteristic TDCA categories of change in sequencing coverage as a function of time. These six categories of behaviour are defined as follows:

1) Rises: Sequencing coverage increases over time and data are modelled to a single 4P sigmoid having a negative IRI.
2) Falls: Sequencing coverage decreases over time and data are modelled to a single 4P sigmoid with a positive IRI.
3) Hills: Sequencing coverage increases and then decreases over time and data are modelled to two 4P sigmoids - a rise then a fall.
4) Valleys: Sequencing coverage decreases and then increases over time and data are modelled to two 4P sigmoids - a fall then a rise.
5) Undefined: Do not display the behavior of the previous categories but are nevertheless modelled as either single rise or fall.
6) Eliminated: Loci that are predicted to behave as a certain category but do not.

TDCA normalizes sequencing coverage data before modelling. This is done in a two fold manner. Firstly, the values at each loci are normalized by the maximum sequencing coverage at non-peak loci for all time points in a time course replicate. Using non-peak loci is meant to capture true background sequencing levels. Table 3 indicates this behaviour in tabular format.

Table 3. Normalization of time course Chip-seq experiments based on sequencing coverage.

Time course experiment (relative time units)	Coverage at non-peak loci	Normalization constant
1	5,000,000	1.20
2	6,000,000	1.00

3	5,500,000	1.09
4	4,000,000	1.50

The normalization constant is unique for each time point and is carried out throughout the program to correct for sequencing coverage. The constant is multiplied by the values calculated at each loci. If users specify the -nonorm flag, all the normalization constant values will be fixed to a value of 1. This can be useful to get an idea of raw coverage values across time points. Additionally, TDCA can incorporate a standard input for normalization. 'Input' refers to sequencing data for an experiment wherein the protein-DNA complexes are not specifically immunoprecipitated and the result is the baseline coverage distribution. If input is provided, the input is normalized by the same manner except using coverage across the entire genome rather than at non-peak loci. The input values for each time point are then subtracted from the experimental coverage values at each loci to a lower limit of zero. If replicates are included, these final values are averaged. However, applying this subtraction strategy may lead to zero inflation depending on the input used. To combat this, we have enabled TDCA to analyze pre-normalized read counts, which allows users to apply the most appropriate normalization strategy to their particular experiment (Liang and Keleş, 2012; Diaz, 2012; Lun and Smyth, 2016). In section 6.4: Creating Normalized Read Counts with DiffBind, we provide an example of how users can achieve normalized read counts using DiffBind (Ross-Innes, 2012), which incorporates the popular programs DESeq2 (Love, 2014) and edgeR (Robinson, 2010) that account for overdispersion. Alternatively, TDCA can use specified normalization values (see -dm flag explanation, section 5.1.9: Reporting Turnover Rates with a Specified Depth Matrix), or pre-normalized read counts depending on the user's preference (see -prenorm flag explanation, section 5.1.10: Reporting Turnover Rates with Pre-Normalized Counts).

TDCA uses normalized coverage values to model data. Without the -model flag, TDCA uses all time points at a given loci to model data which results in either a rise (4P sigmoid increasing in signal over time) or fall (4P sigmoid decreasing in signal over time). If the -model flag is specified, TDCA uses a prediction algorithm based on the time at which the absolute minimum coverage value and absolute maximum coverage value occurs in time. TDCA checks if there are trailing data points (those occurring later in time), and leading data points (those occurring before in time) for the absolute minimum and maximum to decide if a single loci should be modelled as a hill or valley (candidate for double 4P sigmoid modelling). This is where the -s and -t flags come into play.

Given a loci with absolute maximum coverage D_{max} and absolute minimum coverage D_{min}, over time, the range of coverage, $R = D_{max} - D_{min}$. In order to decide if a coverage value at a certain time point is a genuine trailing or leading time point of

the absolute minimum or maximum, the user specified plateau range threshold (-s) value (numerical value *S*, with a range of 0.5-0.95, default = 0.85) is considered.

Genuine leading/trailing data points of the absolute minimum satisfy the equation: $T \geq D_{min} + R * (1 - S)$, and genuine leading/trailing data points of the absolute maximum satisfy the equation: $T \leq D_{max} - R * (1 - S)$, where time point *T* is a leading/trailing data point of the absolute minimum or maximum.

Given the implications of the -s flag, the user can tailor their desire for points to be considered as genuine leading or trailing data points of the absolute maximum and absolute minimum. TDCA is more likely define loci categories as rises and falls rather than hills and valleys as the -s parameter becomes smaller. This is useful for data with significant noise where the expected behaviour is of rises/falls.

The -s flag is essential in determining if a data point is considered a genuine leading or trailing data point of the absolute minimum and maximum coverage values. The leading/trailing threshold (-t) flag also affects how TDCA will define loci categories. -t decides how many of these genuine leading and trailing points (determined by -s) are necessary to define the categorical shift from rises or falls to hills or valleys, essentially determining if a single or double 4P sigmoid should be modelled to the coverage values at a given locus. The default value of -t is 1 (base zero numbering) and is translated into the requirement of at least 2 genuine data points necessary to lead or trail the absolute minimum and/or maximum in order to shift locus modelling from a single 4P to double 4P. The lowest value of -t is 0, which requires only 1 coverage value defined as a genuine leading or trailing value of the absolute minimum or maximum coverage. Figure 3 shows sequencing coverage of a hypothetical locus over 6 time points.

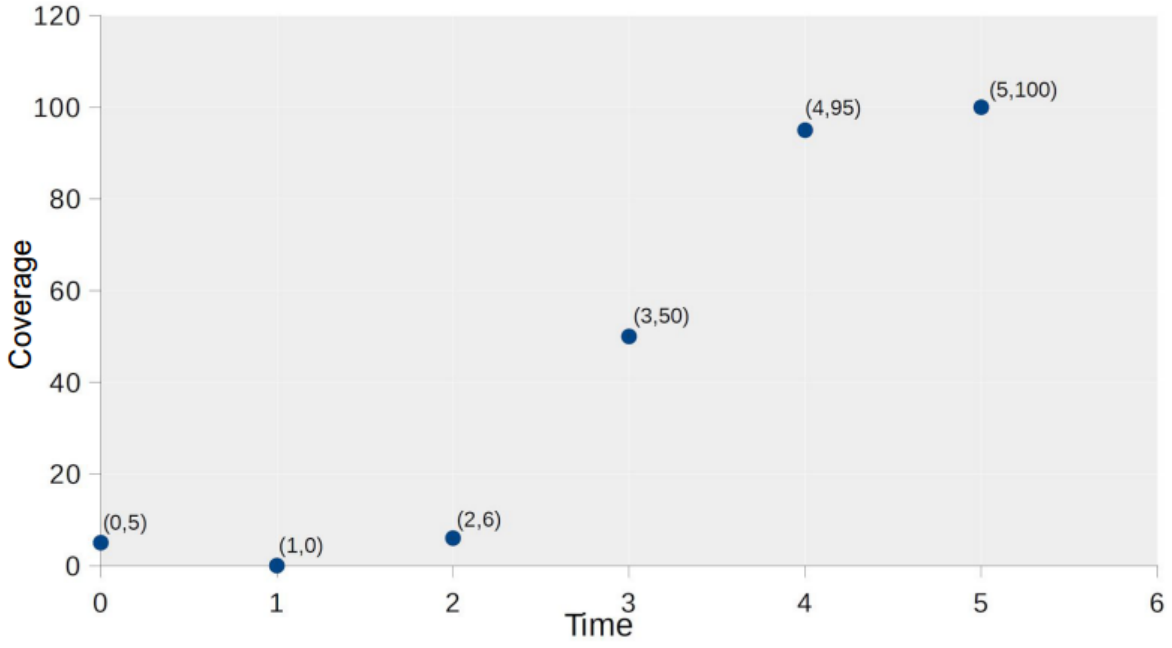


Figure 3. Sequencing coverage of a hypothetical locus over 6 time points. Relative time units are indicated on the x-axis and coverage on the y-axis.

The locus in Figure 3 has a coverage range, *R*, of 100 - 0 = 100. Since this locus has its absolute maximum at the last time point, it cannot have any data points trailing the absolute maximum. In order for the data point (4,95) to be considered a genuine leading data point of the absolute maxima, it must be less than or equal to D_{max} - *R* * (1 - *S*). Solving for *S* results in *S* = 0.95. So, if the -s flag was set to 0.95 or lower, then the data point (4,95) would be considered a genuine trailing data point of the absolute maximum. The remaining leading points would also result in genuine leading data points.

Similarly, for the absolute minimum data point (1,0), the -s flag must be set to a value equal or less than 0.95 for the data point (0,5) to be considered a genuine leading data point. Also, the -s flag must be set to a value equal or less than 0.94 for the data point (2,6) to be considered a genuine trailing data point. Notice how the lower the -s flag is set, the more likely data points will be considered genuine leading or trailing values.

TDCA then determines if data at each locus should be separated into two 4P sigmoid curves based on whether the number of genuine leading and trailing data points of the absolute minimum and maximum exceed the -t flag value. Ultimately, this results in the assignment of a locus as either a rise, fall, hill, valley, or undefined. Loci that are undefined contain data points that are not defined in the categorical prediction algorithm. Undefined loci are, however, still analysed by TDCA as a single 4P sigmoid, resulting in either undefined rises or undefined falls.

Now that TDCA has defined a category for each locus, data is modelled with drc. For data with increasing signal over time (rises, undefined rises, and inclines of hills and valleys), TDCA uses a 4P sigmoid equation as default. If this results in a lower asymptote of less than zero (biologically meaningless since one cannot sequence negative DNA), then TDCA forces the lower asymptote to zero. If, however, the lower asymptote is less than zero and the inflection point is five times greater than the maximum time point (error prone prediction), TDCA forces the lower asymptote to the minimum coverage value. This recovers data closer to its true values based on simulated data testing. For data with decreasing signal over time (falls, undefined falls, and the declines of hills and valleys), TDCA uses a similar procedure as above except the upper asymptote is fixed to the maximum coverage value.

Once modelling is complete, TDCA verifies if the models match the prediction. If not, the locus is eliminated. Note that running TDCA with no model, there cannot be eliminated loci. Figure 4 shows a visual representation of the TDCA core algorithm. The prediction algorithm is shown in Figure 5. The graphical output mainly focuses on a turnover time index (TTI), which is the inflection point adjusted by the asymmetry factor (see equation 1). This is indicative of the binding half life of a protein at a particular locus.

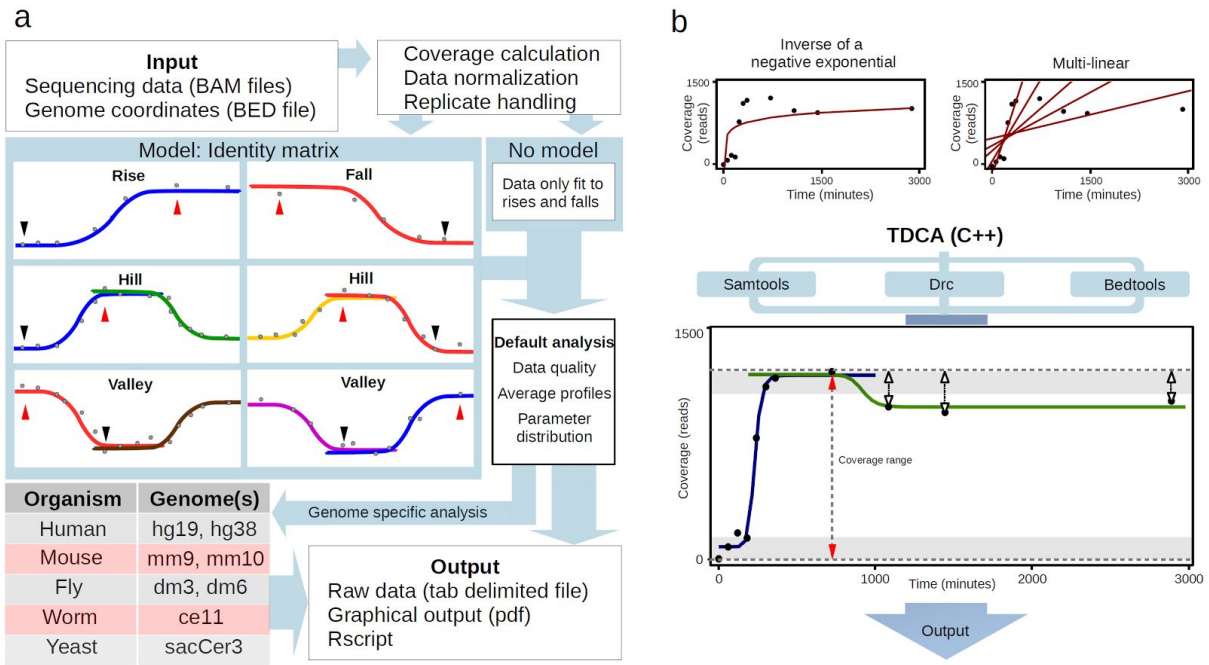


Figure 4. TDCA analysis work flow, requirements, and performance. (**A**) Simplified workflow. Required input data are genomic coordinates in BED format and folders containing BAM TC sequence files. TDCA normalizes data based on total sequencing coverage of each time point and also handles input files and replicates using additional normalization procedures. Loci can be modeled as the following categories of signal change: rise, fall, hill, or valley. An identity matrix that predicts loci category is based on the time at which absolute minimum sequencing coverage (black arrows ▼) and absolute maximum sequencing coverage (red arrows ▲) occurs as set by user defined thresholds. Each sigmoid color indicates a rise or fall with different combinations of absolute maximum and absolute minimum coverage positions in time with genuine leading and trailing points. Alternatively, users can model all their data to a single sigmoidal curve. The resulting parameters from data fitting are then reported to the user along with raw sequencing coverage calculations. Graphical output is provided to the user which can be enriched by specifying genome and genes. R scripts are provided in case users would like to change the look of default figures. (**B**) Plots show sequencing coverage (y-axis) over time (x-axis) at loci for coordinates of chromosome 1:5012338-5013264 obtained from a H3.3 ChIP-seq experiment (Kraushaar, 2013) using previously applied modeling strategies of inverse negative exponential (upper left) and multi-linear (upper right), and the sigmoidal fitting used by TDCA (lower). TDCA requires on terminal access to samtools (Li, 2009) for sequencing coverage calculation of BAM files, bedtools (Quinlan, 2010) for BED file manipulations, and R with the drc (Ritz, 2015) package for curve fitting. In the example shown here, parameters that govern data modeling by TDCA can be fine-tuned to result in either a single or double sigmoid. The lower and upper horizontal dashed lines represent absolute minimum coverage and absolute maximum coverage values, respectively. The overall sequencing coverage

range at a locus is shown as a vertical dashed line with red arrows. In this case, the three data points marked with white arrows exceed the plateau range threshold (gray boxes) and are defined as genuine absolute maximum trailing data points. This results in double sigmoid modeling as shown here. Parameters for both sigmoids are reported to users. The plateau range threshold and leading/trailing threshold could be adjusted such that the locus is modeled to a single sigmoid.

Behaviour	rise	fall	hill		valley	
($T_{min} < T_{max}$)	1	0	1	0	0	1
($T_{min} > T_{max}$)	0	1	0	1	1	0
($T_{min} > T_{max}$) + (T_{min} trail)	0	0	0	0	1	0
($T_{min} < T_{max}$) + (T_{min} lead)	0	0	1	0	0	0
($T_{min} < T_{max}$) + (T_{max} trail)	0	0	0	0	0	1
($T_{min} > T_{max}$) + (T_{max} lead)	0	0	0	1	0	0

Figure 5. TDCA category prediction algorithm. Behaviour colours correspond to sigmoid colours in Figure 4 (a). If a locus satisfies the behaviour statement then there is a boolean true (1) in that row. Hill and valley categories can result in two different ways. If there are loci with a prediction matrix that is not one of the six defined matrices above, then that loci are undefined but are nevertheless modelled by drc as either a rise or a fall. If a locus is predicted to behave as a defined category but is not modelled as such then it is eliminated. T_{min} and T_{max} represent the time at which the absolute minimum and maximum coverage values occur. (T_{min} trail) and (T_{min} lead) indicate that T_{min} has a satisfactory number of genuine trailing and leading data points, respectively, and similarly for T_{max}.

Once normalization and modelling is complete, TDCA reports the raw data in a tab delimited file and uses the modelled parameters to create biologically insightful graphs that provide general information about the user's experiment.

4. General Usage Information

4.1 Supported File Format

4.1.1 BED File Format

The required BED file should contain only three columns. The format for -bed <bed_peaks.BED> is the following:

1. Chromosome, the name of the chromosome
    - Any string. ex. “Chr10”
    - Mandatory column

2. Start, the starting point
    - Any positive integer within the range of the genome of interest. ex. “23507998”
    - Mandatory column
3. End, the ending point
    - Any number that is greater than starting point in above, and is a positive integer within the range of the genome of interest. ex. “23508239”
    - Mandatory column

Do not use any special characters in naming BED files as undefined errors may arise.

4.1.2 BAM File Format

The BAM folder for the -bam <bam_files_folder> flag requires bam files to be sorted and indexed and named with a “XXX_integer.bam” extension, where integer is the time in relative units of the time course experiment. ‘XXX’ should denote the additional file name. Index file should be named XXX_integer.bam.bai. For example, a BAM file could be named: one-hour-treatment_1.bam. Do not use any special characters in naming BAM files and the folder which they are contained as undefined errors may arise.

4.1.3 Text File Format

The required text file for -3d <text_file> is a list of refSeq gene names separated by newlines (/n).

The required text file for -dm <text_file> is a list of integers separated by newlines (/n). The number of integers must equal the total of BAM files to be analysed. The coverage normalization value (in integer format) in the -dm file will be assigned to BAM files based on the order of the BAM folder in the argument list and second by increasing time intervals. For further detail, see section 5.1.9: Reporting Turnover Rates with a Specified Depth Matrix.

5. TDCA Suite

5.1 TDCA

5.1.1 Usage and Options Information

Usage: $ tdca -bed <bed_peaks.BED> -bam <bam_files_folder>

Example: $tdca -bed ChIP-seq.peaks.bed -bam bamFolder -i bamInputFolder -g mm9 -3d gene_list.txt -name exp-name

Table 4. TDCA options and detailed description

Options	**Description**
-bed <bed_peaks.BED> -bam <bam_files_folder> (Mandatory if -prenorm not specified)	BED file followed by BAM files folder. Sequencing coverage of loci in bed_peaks.BED are extracted from each BAM file inside bam_files_folder in order to calculate the turnover rates.
-i <input_bam_files_folder> (Optional)	Input BAM files folder path. Input BAM files are used to normalize data.
-g <genome_name> (Optional)	Generates additional graphs including gene feature boxplot of TTI and ideogram TTI heat map. Supported genomes include: human (hg18, hg19), mouse (mm9, mm10), fly (dm3, dm6), worm (ce11), and yeast (sacCer3).
-3d <text_file> (Optional)	Text file. The -3d flag requires the -g flag. A series of compressed 3D scatter plot of coverage for each gene listed in the given text file is generated as PDF.
-s <0.85> (Optional)	Plateau range threshold (allowable range from 0.5-0.95). Default is 0.85. Value only applies if -model is called.
-t <1> (Optional)	Leading/trailing threshold (allowable range from 0-2). Default is 1. Value must be integer and only applies if -model is called.
-name <exp_name> (Optional)	Rename the output file as user-specific. Default is turnover.exp.
-model (Optional)	Model data to rises, falls, hills, and valleys.
-poisson (Optional)	Data modelled to 3 parameter sigmoidal curve assuming Poisson distributed coverage.
-dm <text_file> (Optional)	Text file containing integers for normalization. The number of integers must equal the total number of BAM files.
-prenorm <text_file> (Mandatory if -bed and -bam not specified)	A pre-normalized count data frame is specified.
-nonorm (Optional)	Read coverage will not be normalized based on sequencing coverage of non-peak loci.

-L5 (Optional)	Data is modelled to a five parameter sigmoid instead of the default 4P.
-proc <integer> (Optional)	Specify the number of processors to use in integer format. Default = maximum number of processors.
-lin (Optional)	Linear regression is performed.

5.1.2 Default Behavior

Using only the mandatory parameters -bed and -bam, TDCA generates output containing three quality charts: pie chart of loci categories (rises, falls, etc.), bar chart of absolute minimum and maximum coverage as a percent occurrence of all loci, and a normalized coverage heatmap across time points. The analysis graphs provided include average read profiles of each loci category, log2 upper asymptote and lower asymptote ratios for incline and declines of hills and valleys, scatter plots of TTI and IRI for data separated by signal increase and signal decrease, distributions of delta coverage by locus type, and distributions of residuals from the different sigmoidal curves. Note that for the residuals to be calculated, at least 5 time points must be provided for a given curve, resulting in 1 degree of freedom. Figures 6-12 show images and descriptions of the default output using data from Kraushaar, *et al*. (2013).

The following command would output the default graphs. The requirement of BAM file format is specified in 4.1.1 BAM File Format. BED file format is described in 4.1.2 BED File Format.

For example (-bed <bed_peaks.BED> -bam <bam_files_folder>):

$tdca -bed ChIP-seq.peaks.bed -bam bamFolder/

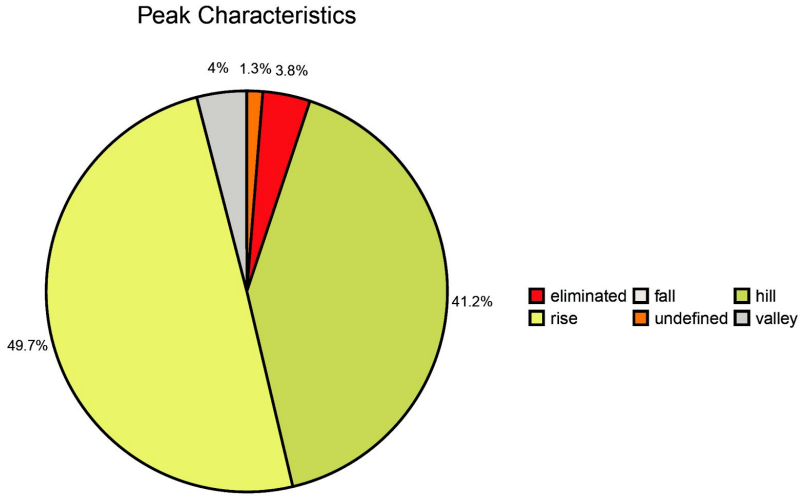


Figure 6. TDCA pie chart of data category. The diagram shows typical display when the -model flag is set. If the data is not modelled, the proportion of only rises and falls will be shown.

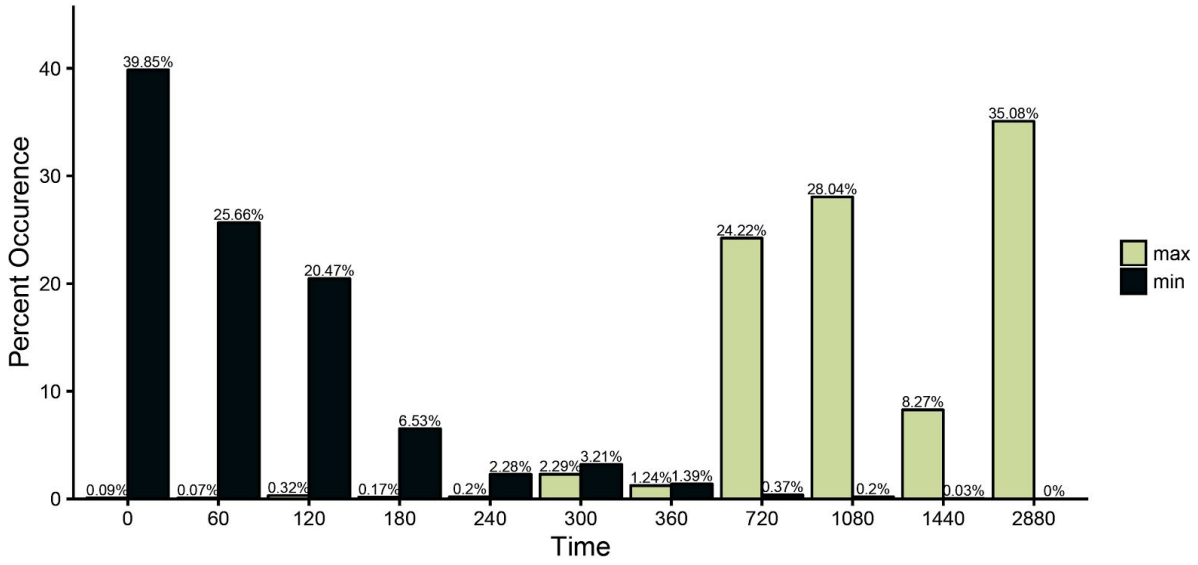


Figure 7. Percent occurrence of absolute minimum and maximum normalized coverage across all loci at each time point. The behaviour shown is typical for data expected to model as rises.

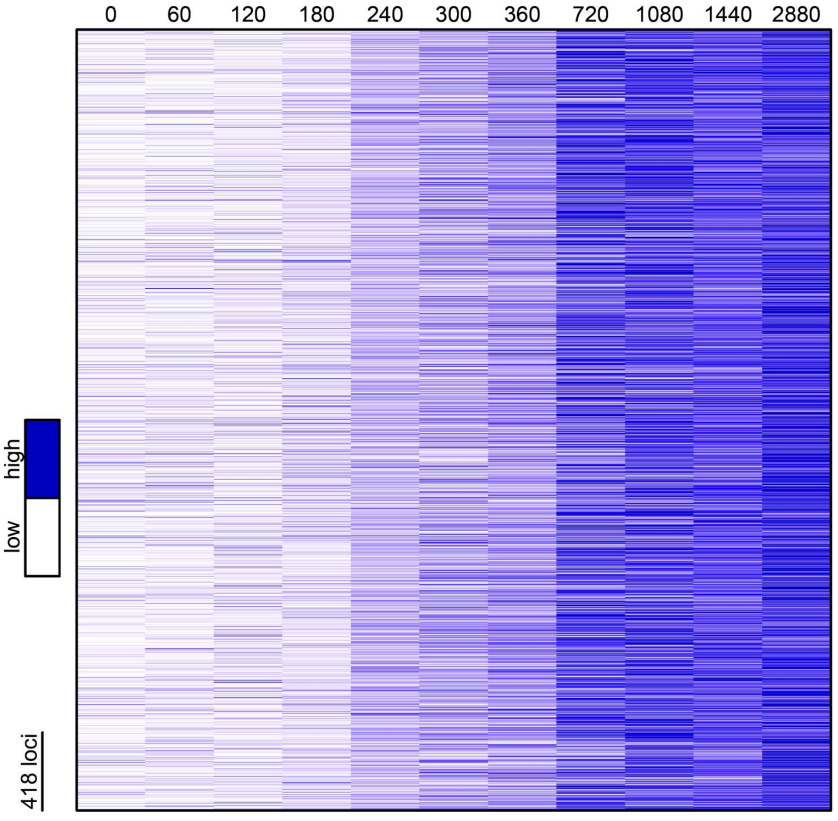


Figure 8. Normalized coverage heatmap across time points. Coverage of each loci across time points is shown as horizontal lines (see scale bar for loci width). Coverage is normalized from 1 (max) to 0 (min) for each locus so that loci can be compared to each other.

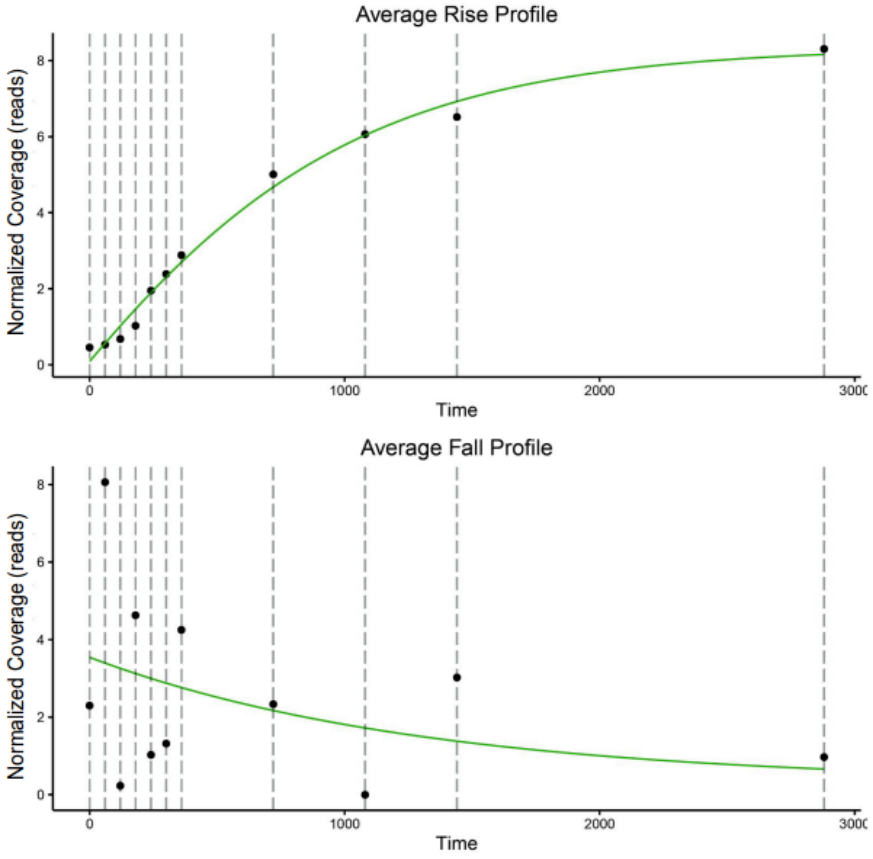


Figure 9. Average rise and fall profiles. TDCA analyses all loci modelled as rises and falls and produces an average profile as a fold enrichment of reads with zero as the lower limit rather than one. The vertical dashed lines indicate the time points used in the analysis. The dots indicate individual averaged values for each time point and the line is the modelled rise or fall. If the data contains no rises or falls then a blank graph will show up with a description indicating so. Falls are messy here because they are not biologically relevant to this particular experiment.

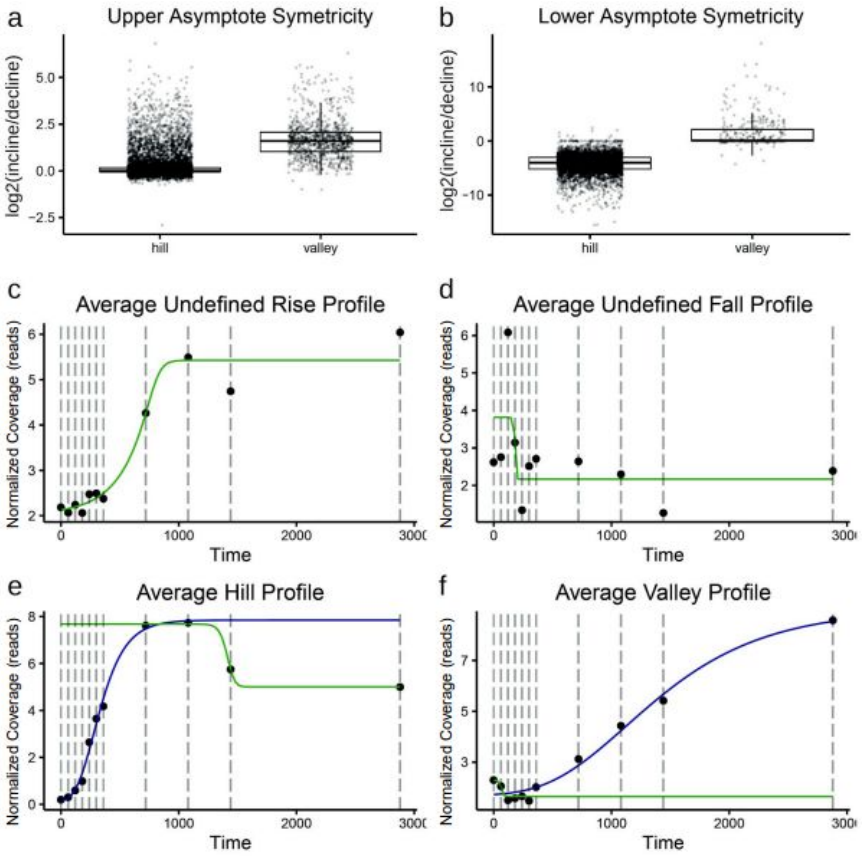


Figure 10. Asymptote symmetricity and loci category profiles. (**A**) log2 upper asymptote ratios for incline and declines of hills and valleys. (**B**) log2 lower asymptote ratios for incline and declines of hills and valleys. The log2 asymptote ratios are an indication of the symmetricity of hills and valleys. (**C-F**) Average profiles for undefined rises, undefined falls, hills, and valleys as in Figure 9. Blue lines in the hills and valleys profiles indicate the model for signal increase (incline) and green lines indicate the model for signal decrease (decline).

For loci that model as hills, TDCA does not ensure that the upper asymptote of hill incline and the upper asymptote of the hill decline will be the same value. Figure 10(A) shows this discrepancy as a log2 ratio of the incline curve over decline curve. This phenomenon is true for both upper and lower asymptotes of hills and valleys. We call this ratio the asymptote symmetricity and graph the values for both upper and lower asymptotes of hills and valleys. In summary, for hills and valleys, TDCA does not use the parameters of one curve to affect the parameters of another curve.

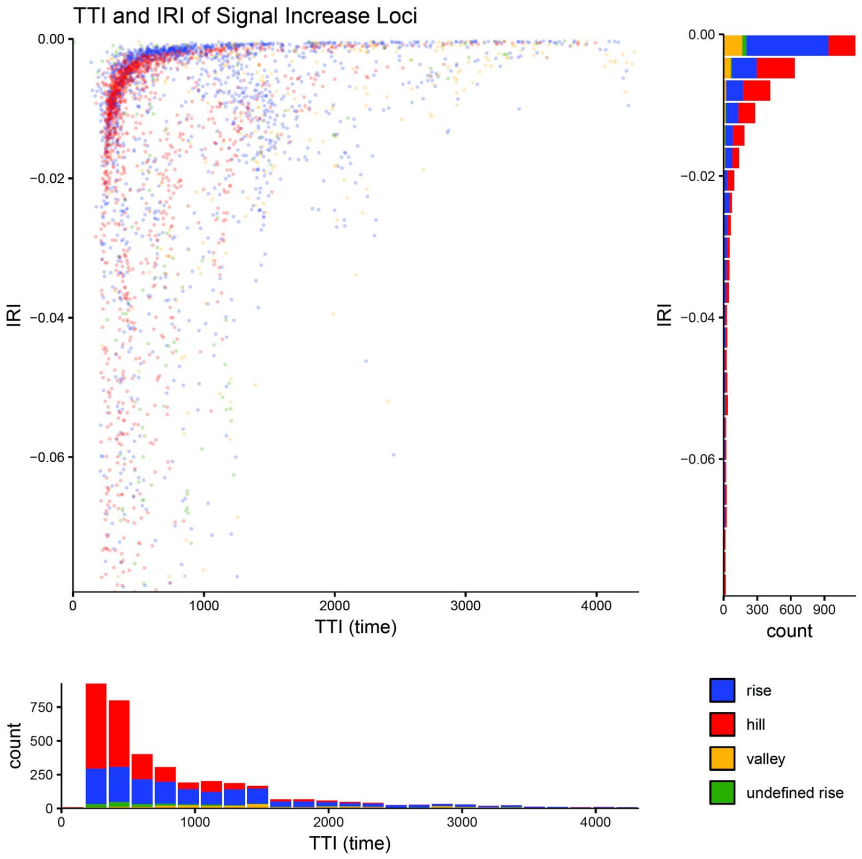


Figure 11. Scatter plot of TTI and IRI for data separated by categories that display signal increase. A separate figure is output for loci that display signal decreases (not shown here). Histograms show loci count, of different loci categories that fall within binned regions of TTI and Hill's coefficient.

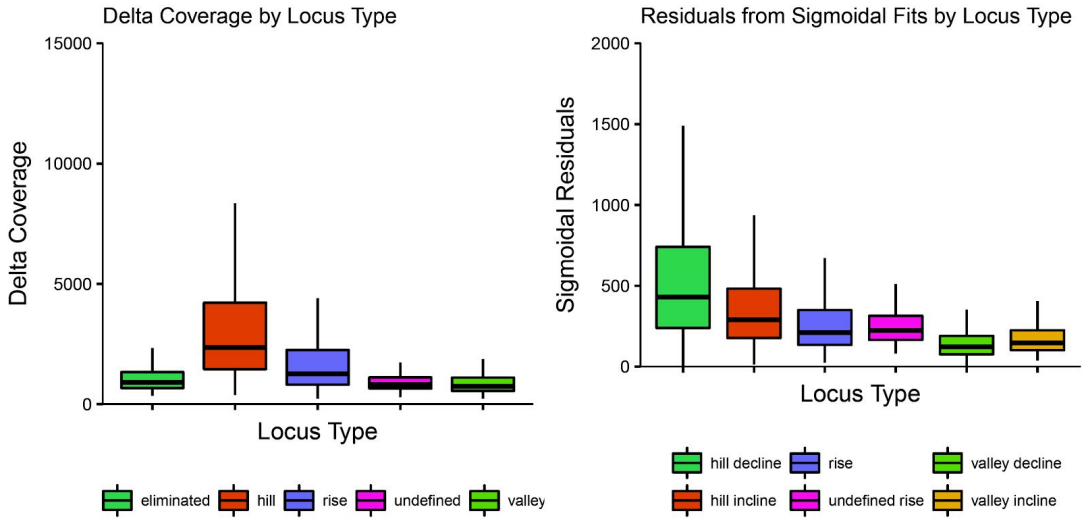


Figure 12. Distribution of delta coverage (bp) by locus type (left) and distribution of residuals from the various sigmoidal curves (right).

5.1.3 Reporting the Turnover Rates with Multiple Replicate BAM Files

With additional replicated BAM files are given by the user, TDCA will take extra time to compute the analysis and graphs. TDCA generates an extra page of graphs for users to perform data quality comparison, shown in Figure 13.

The requirement of BAM file format is specified in 4.1.1 BAM File Format. BED file format is described in 4.1.2 BED File Format.

For example (-bed <bed_peaks.BED> -bam <bam_files_folder_1> -bam <replicated_bam_file_folder_2>):

$tdca -bed ChIP-seq.peaks.bed -bam rep1-bamFolder/ -bam rep2-bamFolder/

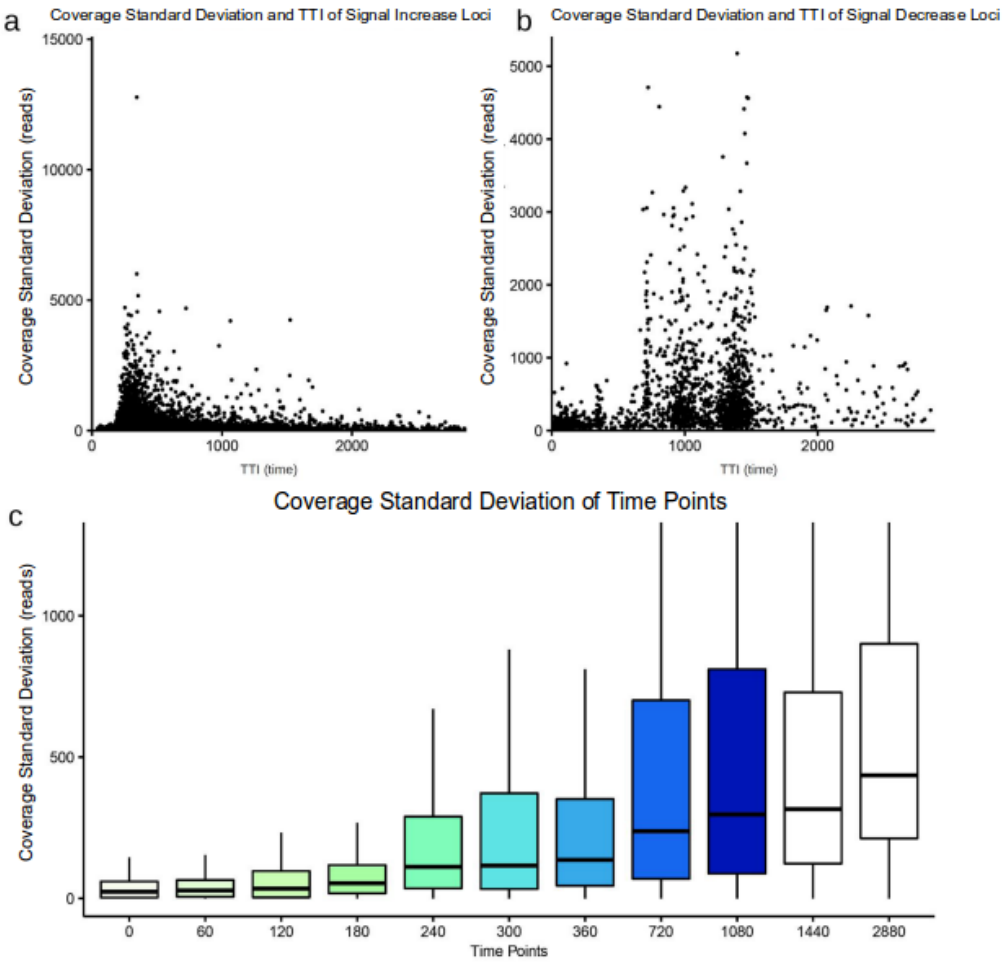


Figure 12. TDCA output with replicates. (**A**) Correlation of standard deviation and upper TTI of loci with signal increase. (**B**) Correlation of standard deviation and upper TTI of loci with signal decrease. (**C**) Distribution of standard deviation at across time points.

5.1.4 Reporting the Turnover Rates of Given BAM Files with Inputs (-i)

Given BAM input files from the user, additional normalization via subtracting coverage of input to a lower limit of zero will be computed. No additional graphs are created, the calculations are simply adjusted accordingly.

The requirement of BAM file format is specified in 4.1.1 BAM File Format.

For example (-bed <bed_peaks.BED> -bam <bam_files_folder> -i <input_bam_files_folder>):

$tdca -bed ChIP-seq.peaks.bed -bam rep1-bamFolder -i rep1-bamInputFolder -bam rep2-bamFolder -i rep2-bamInputFolder

5.1.5 Reporting the Turnover Rates of Given BAM Files with a Specific Genome (-g)

Given a user-specified genome that is supported, TDCA generates a box plot of TTI values at genomic features. Default genome features include: 3'UTR exon and 1000bp upstream of transcriptional start site, 5'UTR exon and 1000bp downstream of transcriptional end site, coding exons, CpG islands, introns, whole gene - characterized as 1000bp upstream of transcriptional start site to 1000bp downstream of transcriptional end site, and intergenic regions - characterized as reciprocal coordinates of whole gene. Additional gene features can be included by the user in a BED file format. This process is described in section 5.1.10 Expanding Genome Feature Libraries. Supported genomes include human (hg19, hg38), mouse (mm9, mm10), fly (dm3, dm6), nematode (ce11), and yeast (sacCer3). Contact the authors to request additional genomes.

In addition, TDCA generates an ideogram heatmap of TTI values at each canonical chromosome. Figure 14 shows the output of TTI at gene features and Figure 15 shows a heatmap ideogram.

For example (-bed <bed_peaks.BED> -bam <bam_files_folder> -g <genome_name>):

$tdca -bed ChIP-seq.peaks.bed -bam bamFolder -i bamInputFolder -g mm9

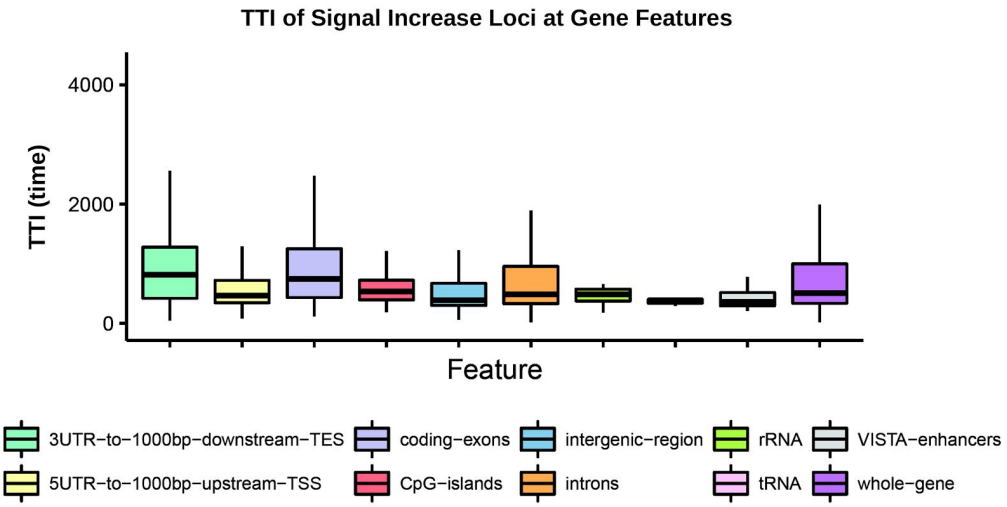


Figure 14. TDCA gene feature boxplot with specified genome. Distribution of TTI values of loci that display signal increase at different gene features. Lower lines, lower part of box, midline, upper part of box, and upper line are 1st quartile, 2nd quartile, median, 3rd quartile and 4th quartile respectively. A boxplot for signal decrease loci is also output (not shown here).

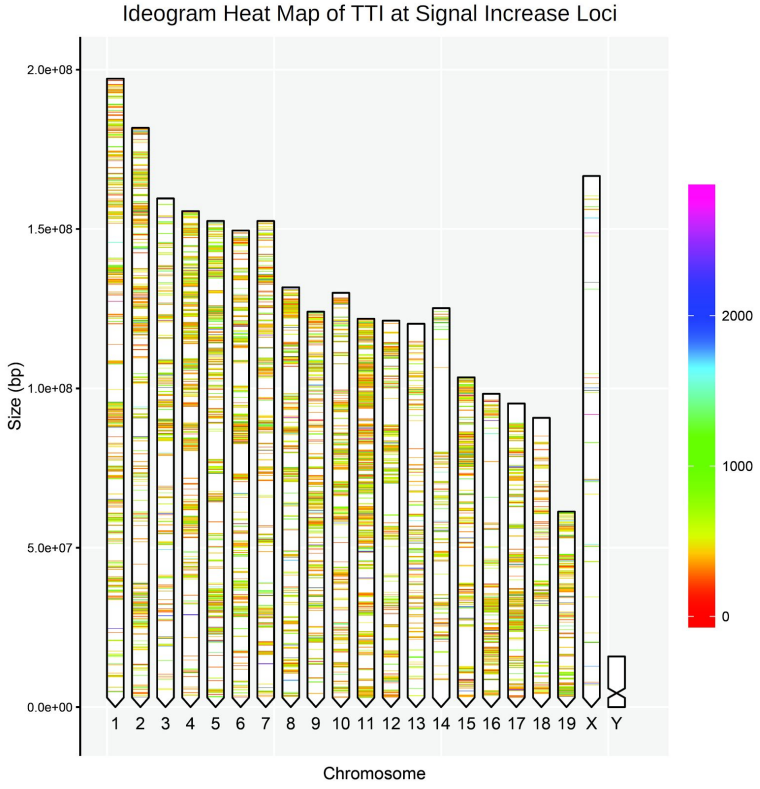


Figure 15. TDCA ideogram heatmap with specified genome. Chromosomes are shown as white polygons with labels at the bottom. Centromeres are located at crosses (most are telometric in mouse - shown here). Bands on the chromosomes indicate where user specified loci are with TTI indicated colorimetrically. Both signal increase loci (shown here) and signal decrease loci (not shown here) are output.

5.1.6 3D Coverage Profiles of User Specified Genes

When a user-specified genome is provided, additional graphs can be printed as a series of 3D scatter plots displaying time dependent coverage for select genes from a user specified gene list. The user-specified gene list is a text file with refSeq gene names separated by newline characters (\n). Figure 16 shows an example of 3D scatter plots at genes.

The required format of the user-specified gene list is described in 4.1.3: Text File Format.

For example (-bed <bed_peaks.BED> -bam <bam_files_folder> -g <genome_name> -3d <text_file>):

$tdca -bed ChIP-seq.peaks.bed -bam bamFolder -i bamInputFolder -g mm9 -3d chr.txt

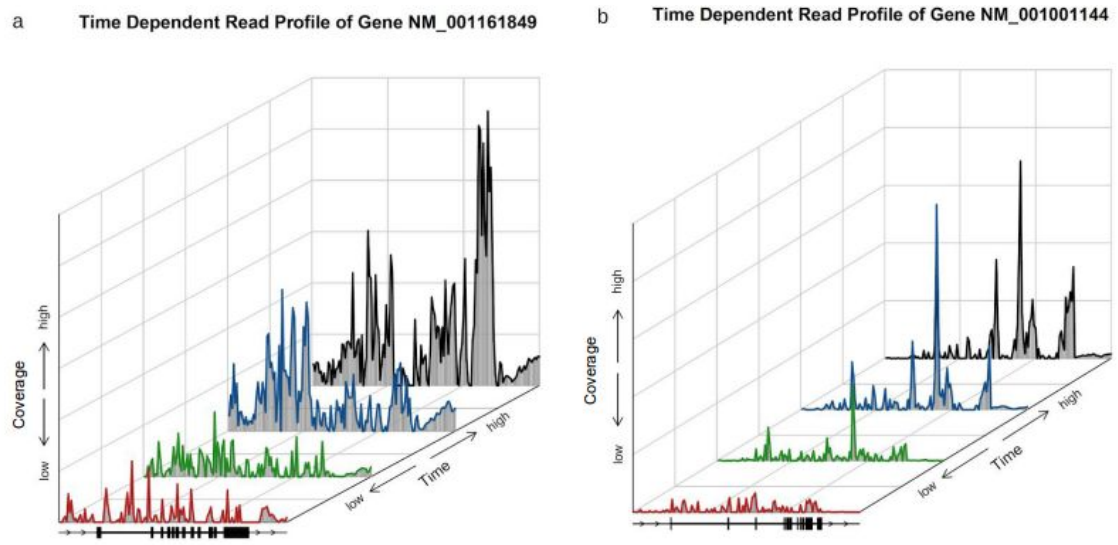


Figure 16. 3D coverage scatter plots of NM_001161849 (a**)** and NM_001001144 (b).

The 3D scatter plot option shows time on the z axis (into the page). Data is compressed to show four time bins. This is done so that the plot is not cluttered. The compression processes is visually described in the Figure 17. The purpose of this diagram is to get a relative idea of turnover times for various peaks located at genes.

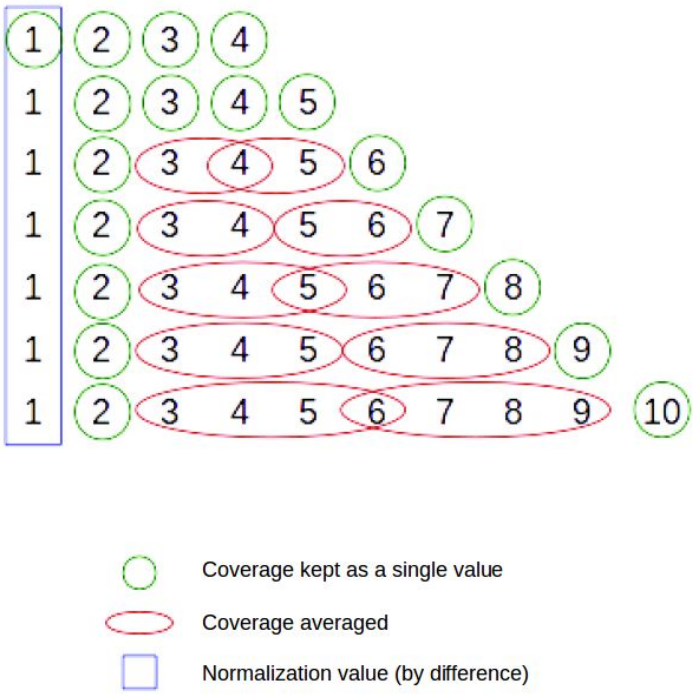


Figure 17. Visual display of 3D scatter plot algorithm. Numbers oriented in horizontal lines represent the number of time points in a given ChIP-seq time course experiment. The amount of time points shown in the scatter plot is set to four no matter how many additional time points are given as can be seen from the green or red circled time points, which represent coverage values that are used in the plot (kept) or average, respectively. The first time point is used as a normalization point by subtracting its coverage from the other data points. An experiment with only four time points would show all four, each normalized by the first, therefore the first time point would look like a flat line.

The x axis of all 3D genes show a picture of the exons in black boxes and introns in thicker black lines. 1000bp upstream and downstream regions of the gene are shown as thinner black lines on the left and right sides of the gene body respectively. An algorithm has been created to search a built in library of refSeq gene information. Keep in mind that refSeq genes have multiple isoforms of certain genes. Remember to choose the appropriate isoform. USCS browser is a great tool to visually inspect coordinates of isoforms (Kent,W.J. *et al*. 2002).

5.1.7 Reporting Turnover Rates with Different Plateau Range Threshold

TDCA offers a plateau range threshold flag (-s). This threshold can be adjusted by the user to be between 0.55-0.95 (default = 0.85). The -s flag is discussed in section 3: Core Algorithm Description. -s plays a major role in the modelling procedure.

5.1.8 Reporting Turnover Rates with Different leading/trailing Threshold

TDCA offers a leading/trailing threshold flag (-t). This threshold can be adjusted by the user to be between 0-2 (default = 1). The -t flag is discussed in section 3: Core Algorithm Description. -t plays a major role in the modelling procedure.

5.1.9 Reporting Turnover Rates with a Specified Depth Matrix

As a response to novel ChIP-seq normalization strategies, such as internal spike in standards, users can specify their own values to normalize BAM files by, genome wide and at non-peak loci, for input and experiment files respectively. A newline(\n) separated file containing integers equal to the number of BAM files must be specified after the -dm flag. TDCA will assign these values to BAM files in order of replicates, input before experiment, and then by chronological order. The user can double check if TDCA assigned the correct values to each file by inspecting the runtime output.

5.1.10 Reporting Turnover Rates with Pre-Normalized Counts (-prenorm)

Depending on the complexity of a given time course sequencing experiment, users may want to take advantage of normalization strategies not offered by TDCA yet are keen to use the automated modelling provided by TDCA. We have included the -prenorm flag for this purpose. Users can normalize their time course data anyway they would like and provide a tab delimited coverage file with a header to TDCA after specifying the -prenorm flag.

The required format of the user-specified pre-normalized count file is described in 4.1.3: Text File Format.

For example (-prenorm <norm_counts.txt>):

$tdca -prenorm norm_counts.txt

The first three columns in norm_counts.txt in the example above would be the same as a BED file (tab delimited: chromosome, start, end). The header of these first three column must be “chromosome”, “start”, “end”. Any additional columns must be the normalized coverage data at each locus across time (tab delimited). The headers of these columns must be a time point in integer format. For example, the first few rows of a normalized count table with 5 time point (1-5) could look like this (2 periods represent normalized counts):

Chromosome		start	end	1	2	3	4	5
2L	500	600	..	..	..	..	..	..	..
2L	1500	1600	..	..	..	..	..	..	..
2L	5500	5600	..	..	..	..	..	..	..

5.1.11 Performing Linear Regression (-lin)

Specifying the -lin flag enables one to perform linear regression as well as the sigmoidal fitting on a time course data set. TDCA will output an additional tab delimited file with the locus coordinates (chromosome, start, end), as well as the y-intercept, slope and residuals from the linear regression. When performing linear regression, TDCA uses the “lm” function in R, using the coverage for all time points (no separation of time points in performed as in the sigmoidal modelling). When the -lin flag is specified, additional graphical outputs are given as shown in Figure 18.

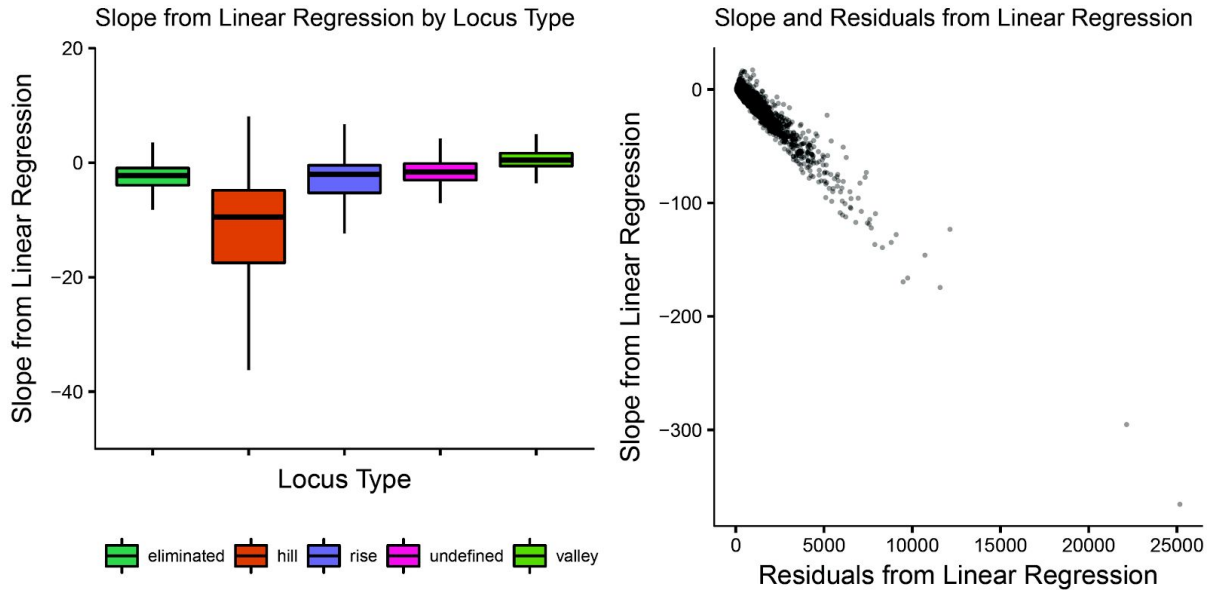


Figure 18. Graphical output from linear regression. Distribution of slope values by locus category (left). Scatter plot of slope and the residuals from the linear regression fit (right).

5.1.12 Reporting Turnover Rates with Poisson Distributed Data (-poisson)

To accommodate a wider range of data distributions, we have enabled TDCA to model time course sequencing coverage at a set of loci to a three parameter Poisson model, using the drm function L.3(), type=poisson function within drc. This strategy puts less weight on time points containing high read counts and may be

more appropriate for certain types of experiments. To implement the Poisson distribution modelling, specify the -poisson flag in combination with other options.

For example (-bed <bed_peaks.BED> -bam <bam_files_folder> -poisson):

$tdca -bed ChIP-seq.peaks.bed -bam bamFolder -poisson

5.1.13 Expanding Genome Feature Libraries

Users can input their own BED file format genome features into the appropriate genome folder located in the TDCA GenomeFeatures folder. UCSC table browser was used to get default libraries (Karolchik D., *et al*. 2004). TDCA will use the newly input file(s) in analysis of TTI at genome features.

6. Example Usage

6.1 Comprehensive Example

In this short subsection we provide users comprehensive usage of TDCA including visual representation of the required files. The following command runs TDCA with 1 replicate, genes for 3D analysis, a depth-matrix, and specification of the mouse mm9 genome:

$tdca -bed loci.bed -bam rep1 -dm depth-matrix.txt -3D genes.txt -g mm9

Figure 19 shows a visual representation of required and additional files for this command.

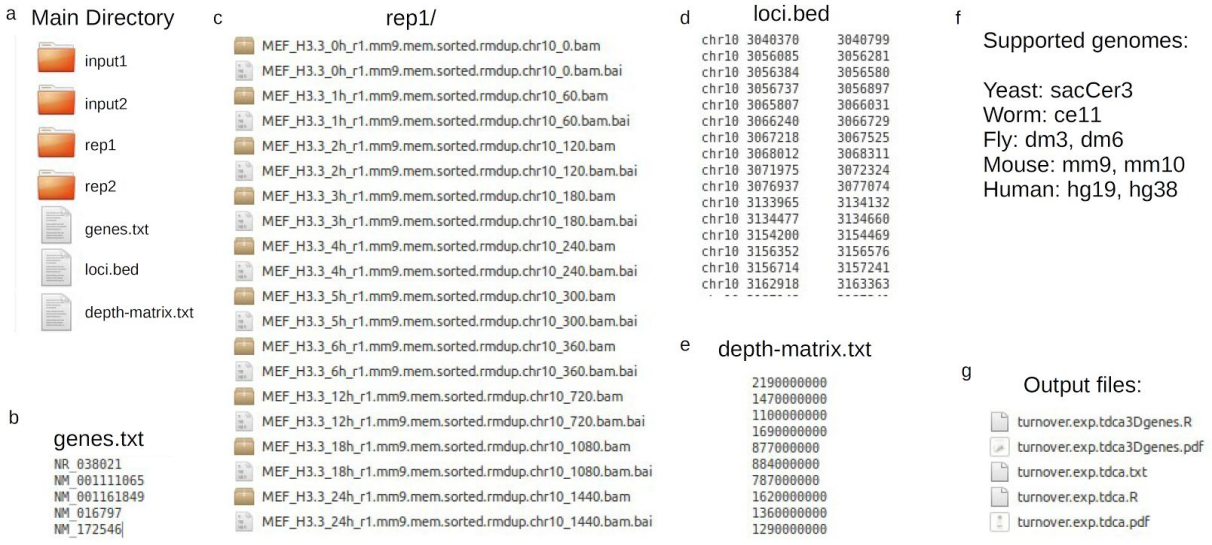


Figure 19. Visual display of directory contents and files for TDCA input. (**A**) Contents of main directory where the user will run TDCA. The folders (rep1, rep2, input1, and input2) contain BAM files, appropriately named, that will be read by TDCA. (**B**) Contents of the genes.txt file: RefSeq ID of five genes separated by a newline (\n) character. (**C**) Contents of the rep1 folder: 10 Bam files properly named

(XXX_time.bam) with indices. The contents of the BAM folders rep2, input1, and input2 must contain the same number of BAM files with the same time points. (**D**) BED file containing 3 tab delimited column, chromosome, start, and end. There can be any number of loci here. (**E**) Depth matrix used to specify values to normalized BAM files to. There must be an equal number of integers separated by a newline (\n) character as there are BAM files in each folder. The integers are assigned to BAM files in chronological order. (**F**) Genomes supported by TDCA -g flag. (**G**) Hypothetical output files generated by TDCA. Along with these is a data folder that the Rscript reads in order to generate a pdf.

If the user wanted to include multiple replicates and input, such as:

$tdca -bed loci.bed -bam rep1 -i input1 -bam rep2 -i input2 -dm depth-matrix.txt -3D genes.txt -g mm9

The depth-matrix.txt file must contain the appropriate number of integers. Alternatively, the user can specify no depth matrix.

6.2 Getting data

During development, TDCA was tested on many published time dependent sequencing data sets. One of the most robust experiments was ChIP-seq performed using an inducible HA tagged H3.3. The GEO accession number for project is GSE51505 (found at: http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE51505). The experiment was done in MEF cells using 2 replicates on 11 time points, including input. In this next section we show users how to get this data and prepare it for TDCA processing. Alternatively, at the end of the section we offer users a link to this pre-processed data so that users can go straight to testing TDCA.

Accession numbers and names of data we will use for example:

GSM1246648	MEF_H3.3_0h_r1
GSM1246649	MEF_H3.3_1h_r1
GSM1246650	MEF_H3.3_2h_r1
GSM1246651	MEF_H3.3_3h_r1
GSM1246652	MEF_H3.3_4h_r1
GSM1246653	MEF_H3.3_5h_r1
GSM1246654	MEF_H3.3_6h_r1
GSM1246655	MEF_H3.3_12h_r1
GSM1246656	MEF_H3.3_18h_r1
GSM1246657	MEF_H3.3_24h_r1
GSM1246658	MEF_H3.3_48h_r1
GSM1246659	MEF_H3.3_72h_r1
GSM1246660	MEF_H3.3_0h_r2
GSM1246661	MEF_H3.3_1h_r2
GSM1246662	MEF_H3.3_2h_r2

GSM1246663	MEF_H3.3_3h_r2
GSM1246664	MEF_H3.3_4h_r2
GSM1246665	MEF_H3.3_5h_r2
GSM1246666	MEF_H3.3_6h_r2
GSM1246667	MEF_H3.3_12h_r2
GSM1246668	MEF_H3.3_18h_r2
GSM1246669	MEF_H3.3_24h_r2
GSM1246670	MEF_H3.3_48h_r2
GSM1246671	MEF_H3.3_0h_Input
GSM1246672	MEF_H3.3_1h_Input
GSM1246673	MEF_H3.3_2h_Input
GSM1246674	MEF_H3.3_3h_Input
GSM1246675	MEF_H3.3_4h_Input
GSM1246676	MEF_H3.3_5h_Input
GSM1246677	MEF_H3.3_6h_Input
GSM1246678	MEF_H3.3_12h_Input
GSM1246679	MEF_H3.3_18h_Input
GSM1246680	MEF_H3.3_24h_Input
GSM1246681	MEF_H3.3_48h_Input
GSM1246682	MEF_H3.3_72h_Input

Download the above SRA experiments and unpack using sra toolkit (Leinonen,R. *et al*. 2011) fastq-dump command. Data then needs to be aligned (Li H and Durbin R. 2009) to the mouse genome and peaks need to be called (Zhang,Y. *et al*. 2008). We used the following commands for this:

Align to mm9:
$bwa mem mm9.fa MEF_H3.3_0h_r1 > MEF_H3.3_72h_r1.mm9.mem_0.sam
Convert sam to bam:
$samtools view -bS MEF_H3.3_72h_r1.mm9.mem_0.sam >
MEF_H3.3_72h_r1.mm9.mem_0.bam
Sort bam:
$samtools sort MEF_H3.3_72h_r1.mm9.mem_0.bam
MEF_H3.3_72h_r1.mm9.mem.sorted_0.bam
Remove duplicates if any:
$samtools rmdup MEF_H3.3_72h_r1.mm9.mem.sorted_0.bam
MEF_H3.3_72h_r1.mm9.mem.sorted.rmdup_0.bam
Create bam index:
$samtools index MEF_H3.3_72h_r1.mm9.mem.sorted.rmdup_0.bam
MEF_H3.3_72h_r1.mm9.mem.sorted.rmdup_0.bam.bai

Repeat this for each fastq file. Note that the bam files are named with the convention “XXX_integer.bam”. This is naming convention is essential for TDCA to detect the time point in question. TDCA uses regular expression to do this. The integer extension for time course BAM files should all be in the same units. Next, call peaks using time point with longest treatment time: 4320 minutes (72 hours of doxycycline treatment).

Call broad peaks with macs2 (77531 peaks):

$macs2 callpeak -t MEF_H3.3_72h_r1.mm9.mem.sorted.rmdup_4320.bam -c MEF_H3.3_72h_input.mm9.mem.sorted.rmdup_4320.bam --broad -g mm -name h3.3.72h-72hinput.macs2-broad.0.05 --broad-cutoff 0.05

The bed file required for TDCA input must contain 3 tab delimited columns: chromosome, start, and end for each peak. Copy and paste these into a text file from the macs2 xls output. Once data is obtained, put all the bam files and indices with correct name extension for time points (XXX_integer.bam) in a folder for each replicate and input.

For example:
Make a directory for replicate 1 files:
$mkdir Kraushaar-rep1
Move files replicate 1 files to newly created directory:
$mv -t ./Kraushaar-rep1 MEF_H3.3_48h_r1.mm9.mem.sorted.rmdup_2880.bam
MEF_H3.3_72h_r1.mm9.mem.sorted.rmdup.bam
MEF_H3.3_0h_r1.mm9.mem.sorted.rmdup_0.bam
MEF_H3.3_1h_r1.mm9.mem.sorted.rmdup_60.bam
MEF_H3.3_3h_r1.mm9.mem.sorted.rmdup_180.bam
MEF_H3.3_5h_r1.mm9.mem.sorted.rmdup_300.bam
MEF_H3.3_12h_r1.mm9.mem.sorted.rmdup_720.bam
MEF_H3.3_24h_r1.mm9.mem.sorted.rmdup_1440.bam
MEF_H3.3_48h_r1.mm9.mem.sorted.rmdup_2880.bam
MEF_H3.3_6h_r1.mm9.mem.sorted.rmdup_360.bam
MEF_H3.3_18h_r1.mm9.mem.sorted.rmdup_1080.bam
MEF_H3.3_2h_r1.mm9.mem.sorted.rmdup_120.bam
MEF_H3.3_4h_r1.mm9.mem.sorted.rmdup_240.bam
MEF_H3.3_72h_r1.mm9.mem.sorted.rmdup.bam
MEF_H3.3_0h_r1.mm9.mem.sorted.rmdup_0.bam.bai
MEF_H3.3_24h_r1.mm9.mem.sorted.rmdup_1440.bam.bai
MEF_H3.3_4h_r1.mm9.mem.sorted.rmdup_240.bam.bai
MEF_H3.3_12h_r1.mm9.mem.sorted.rmdup_720.bam.bai
MEF_H3.3_2h_r1.mm9.mem.sorted.rmdup_120.bam.bai

MEF_H3.3_5h_r1.mm9.mem.sorted.rmdup_300.bam.bai
MEF_H3.3_18h_r1.mm9.mem.sorted.rmdup_1080.bam.bai
MEF_H3.3_3h_r1.mm9.mem.sorted.rmdup_180.bam.bai
MEF_H3.3_6h_r1.mm9.mem.sorted.rmdup_360.bam.bai
MEF_H3.3_1h_r1.mm9.mem.sorted.rmdup_60.bam.bai
MEF_H3.3_48h_r1.mm9.mem.sorted.rmdup_2880.bam.bai
MEF_H3.3_72h_r1.mm9.mem.sorted.rmdup_4320.bam.bai

Repeat this for both replicates and input. If the user is working with the two replicates and input from Kraushaar et al. (2013) then there should be three folders corresponding to the two replicates and the input, each containing appropriately named files with indices. The working directory should also contain a BED file of H3.3 peaks.

$ls
Kraushaar-rep1
Kraushaar-rep2
Kraushaar-input
H3.3.72h-72hinput.macs2-broad.0.05.chr10.bed

Note that the H3.3 data only has time point 4320 for replicate 1. TDCA will give an error if replicates have differing timepoints. So the name of the bam file for time point 4320 was changed from MEF_H3.3_72h_r1.mm9.mem.sorted.rmdup_4320.bam to MEF_H3.3_72h_r1.mm9.mem.sorted.rmdup.bam. The absence of the “XXX_integer naming convention will make it invisible to TDCA.

Alternatively, we provide pre-aligned chromosome 10 data along with peaks for faster testing. This data can be found here: https://drive.google.com/open?id=0B5BFPUpdPrmhdG5jbkFUSlEtZDA

6.3 Running TDCA

These files now satisfy the basic input requirement to run TDCA. TDCA commands could be ran as follows:

Run replicate 1 with no genome specified (heatmap ideogram and gene features boxplot will not be created):
$tdca -bed H3.3.72h-72hinput.macs2-broad.0.05.chr10.bed -bam Kraushaar-chr10-rep1 -name r1.chr10.minimum

Run both replicates and input and specify genome:

$tdca -bed H3.3.72h-72hinput.macs2-broad.0.05.chr10.bed -bam Kraushaar-chr10-rep1 -bam Kraushaar-chr10-rep2 -i Kraushaar-chr10-input -i Kraushaar-chr10-input -g mm9 -name r1.r2.input.chr10.mm9

6.4 Creating Normalized Read Counts with DiffBind

As mentions, users may wish to normalize their data using other published techniques and provide the normalized counts to TDCA for sigmoidal fitting to gain the turnover parameters presented previously. In this section, we provide a brief example of how a user can do this using DiffBind. The text below in blue represents the R code required to perform such an analysis. The sample sheet named: 'diffbind_samplesheet.csv' in the code below should include metadata regarding the BAM files that the user would like to analyse. A sample sheet with two replicates of a time course experiment with five time points and a control for each replicate with input control for each BAM file might look something like this:

```
SampleID,Tissue,Factor,Condition,Treatment,Replicate,bamReads,ControlID,bamControl,Peaks,PeakCaller
r1_0,HEK,0h,wt,timecourse,1,bams/r1.0.bam,input.r1_0,inputBams/input.r1.0.bam,peaks.bed,bed
r1_1,HEK,1h,wt,timecourse,1,bams/r1.1.bam,input.r1_1,inputBams/input.r1.1.bam,peaks.bed,bed
r1_2,HEK,2h,wt,timecourse,1,bams/r1.2.bam,input.r1_2,inputBams/input.r1.2.bam,peaks.bed,bed
r1_3,HEK,3h,wt,timecourse,1,bams/r1.3.bam,input.r1_3,inputBams/input.r1.3.bam,peaks.bed,bed
r1_4,HEK,4h,wt,timecourse,1,bams/r1.4.bam,input.r1_4,inputBams/input.r1.4.bam,peaks.bed,bed
r1_5,HEK,5h,wt,timecourse,1,bams/r1.5.bam,input.r1_5,inputBams/input.r1.5.bam,peaks.bed,bed
r1_5,HEK,control,wt,control,1,bams/r1.control.bam,input.r1_control,inputBams/input.r1.control.bam,peaks.bed,bed
r2_0,HEK,0h,wt,timecourse,2,bams/r2.0.bam,input.r2_0,inputBams/input.r2.0.bam,peaks.bed,bed
r2_1,HEK,1h,wt,timecourse,2,bams/r2.1.bam,input.r2_1,inputBams/input.r2.1.bam,peaks.bed,bed
r2_2,HEK,2h,wt,timecourse,2,bams/r2.2.bam,input.r2_2,inputBams/input.r2.2.bam,peaks.bed,bed
r2_3,HEK,3h,wt,timecourse,2,bams/r2.3.bam,input.r2_3,inputBams/input.r2.3.bam,peaks.bed,bed
r2_4,HEK,4h,wt,timecourse,2,bams/r2.4.bam,input.r2_4,inputBams/input.r2.4.bam,peaks.bed,bed
r2_5,HEK,5h,wt,timecourse,2,bams/r2.5.bam,input.r2_5,inputBams/input.r2.5.bam,peaks.bed,bed
r2_5,HEK,control,wt,control,2,bams/r2.control.bam,input.r2_control,inputBams/input.r2.control.bam,peaks.bed,bed
```

The R code could be executed as follows:

```
# Diffbind tutorial. For more information see the following:
# http://genomicsclass.github.io/book/pages/ChIPseq.html
# http://bioconductor.org/packages/release/bioc/manuals/DiffBind/man/DiffBind.pdf
library(DiffBind)
# set directory to where BAM files are located
setwd("/DiffBind_Directory")
ta <- dba(sampleSheet="diffbind_samplesheet.csv") # meta data here
ta2 <- dba.count(ta, minOverlap=3)
# this gives a heatmap of normalized count similarities:
plot(ta2)

ta2 <- dba.contrast(ta2, categories=DBA_TREATMENT, minMembers=2)
ta2 <- dba.analyze(ta2)
```

```
#Retrieve all sites with confidence stats and normalized counts
norm_counts <- dba.report(ta2, th=1, bCounts=TRUE)
write.table(norm_counts, "/DiffBind_Directory/DiffBind-norm-counts.txt", sep="\t")

#Retrieve all sites with confidence stats and raw counts
raw_counts <- dba.report(ta2, th=1, bCounts=TRUE,bNormalized=FALSE)
write.table(raw_counts, "/DiffBind_Directory/DiffBind-raw-counts.txt", sep="\t")
```

The user could then tailor the output files called 'DiffBind-raw-counts.txt' and 'DiffBind-norm-counts.txt' according to the -prenorm file input specifications.

7. FAQ

6.1 Installation fails for R package, "rgl" in Ubuntu environment because of X11 not found but required.

Run the following in command line, sudo apt-get install r-cran-rgl.

6.2 Generate pdf file without compiling tdca.

Once the R scripts are generated by tdca, run the following in command line, Rscript name_R_script. "xxx.tdca3Dgenes.R" is used to generate 3D graphs and "xxx.tdca.R" generates default graphs

6.3 Changing the look of output graphs.

R scripts are provided for each graphical output. Users may wish to change the look of certain graphs and can do so with a basic understanding or R and ggplot2. R scripts are generated in a modular fashion to facilitate single change options. Data generated from larger genomes may create extremely large R scripts and may not open well with all text editors. The format of all R scripts is the same given the same tdca flag calls. Thus R scripts can also be manipulated by line swapping using combinations of awk and cat or other commands.

6.4 What compiler do I need to install tdca?

TDCA is compiled using g++. Most later versions should work and version 4.9.3 has been tested exhaustively. C++ standard library 2014 (-std=c++14) is used in the TDCA Makefile, however C++ standard library 2011 also works. Users may change the -std=c++14 flag to -std=c++11 in the TDCA Makefile if they wish. The openmp library used for parallelization requires an appropriate compiler.

6.5 Sometimes while TDCA is running the drc script, the following error is produced:

Error in optim(startVec, opfct, hessian = TRUE, method = optMethod, control = list(maxit = maxIt, : non-finite finite-difference value [1]

Error in drmOpt(opfct, opdfct1, startVecSc, optMethod, constrained, warnVal, : Convergence failed

What does this mean?

This error is produced when drc cannot model a locus. TDCA catches this error and simply classifies the locus as eliminated.

8. Runtime Dependencies

8.1 Number of Processors/BED File Peaks/ BAM files

TDCA is parallelized to run on all available processors. Runtime dependencies were tested using replicates from Kraushaar et al. (2013) on the bugaboo server of westgrid computer cluster.

TDCA outputs the coverage at non-peak loci for experiment files and the total coverage of input files. If only 1 processor is in use, the order that the coverage of the files are printed will be chronological. If, however, openmp is enabled, the files will be printed in an unpredictable way. This is one simple way to check if parallelization is working. Secondly, if parallelized, these files will be printed in clusters, rather than one by one in sequential blocks. When TDCA is running the drc R script, the user can check the directory that the program is running if there are R script named drcVersitile.X.R, where X is an integer, equal to the number of processors then parallelization is working. These are simple ways to ensure parallelization is working. Using openmp on cloud clusters may require additional efforts to ensure proper functioning. For example, on westgrid systems it is required that parallel jobs be ran on processors on the same node. Furthermore, it is best to run jobs in seperate folders in case of hard coded file crosstalk. As shown in Figure 20, the run time of TDCA decreases as increased processing power is available.

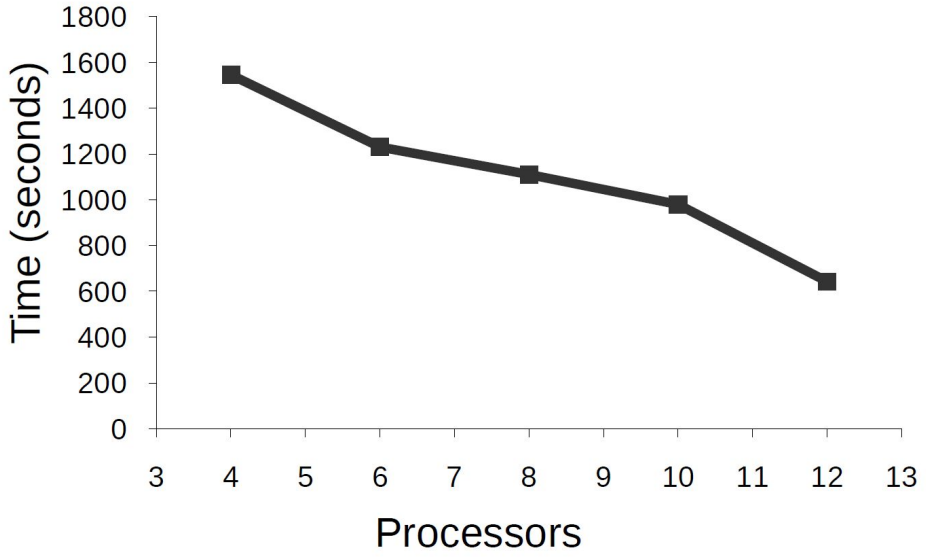


Figure 20. Processing time of H3.3 time course data using eleven time points on chromosome 10 loci (4180 loci) with 4, 6, 8, 10, and 12 processors. TDCA utilizes openmp to parallelize various algorithms in the program.

It is worth mentioning that if the -prenorm flag is used, the samtools depth command is skipped, and therefore the runtime is decreased. If users find themselves running TDCA over and over, a count file can be created from the output file, which will minimize repeat analysis time.

9. TDCA Support

Please submit bug reports and request for library expansions to: mmyschyshyn@gmail.com

10. References

Deal, R.B. *et al*. (2010) Genome-wide kinetics of nucleosome turnover determined by metabolic labeling of histones. *Science*, **328**, 1161-4.

Diaz, A. *et al*. (2012) Normalization, bias correction, and peak calling for ChIP-seq. *Stat. Appl. Genet. Mol. Biol.* **11**, 9.

Karolchik D., *et al*. (2004) The UCSC Table Browser data retrieval tool. *Nucleic Acids Res.* **1**, D493-6.

Kent,W.J. *et al*. (2002) The human genome browser at UCSC. *Genome Res.***12**, 996-1006.

Kraushaar, D.C. *et al*. (2013) Genome-wide incorporation dynamics reveal distinct categories of turnover for the histone variant H3.3. *Genome Biol.*, **14**, R121.

Leinonen, R. *et al*. (2011) The Sequence Read Archive. *Nucleic Acids Res*. **39**, D19-D21.

Li H and Durbin R. (2009) Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics* **15**, 1754-60.

Li, H. *et al.* (2009) The Sequence alignment/map (SAM) format and SAMtools. *Bioinformatics*, **25**, 2078-9.

Liang, K. and Keleş, S. (2012) Normalization of ChIP-seq data with control. BMC *Bioinformatics*, **13**, 199.

Love, M.I. *et al*. (2014) Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. *Genome Biol.* **15**, 550.

Lun, A.T.L. *et al*. (2016) csaw: a Bioconductor package for differential binding analysis of ChIP-seq data using sliding windows. *Nucleic Acids Res.* **44**, e45.

Wickham,H. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York, 2009.

Quinlan, A.R. and Hall, I.M. (2010) BEDTools: a flexible suite of utilities for comparing genomic features. *Bioinformatics*, **26**, 841-842.

Ritz, C. *et al*. (2015) Dose-Response Analysis Using R. *PLos One*, **10**, e0146021.

Robinson, M.D. *et al*. (2010) edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics*, **26**, 139-40.

Ross-Innes, C.S. *et al*. Differential oestrogen receptor binding is associated with clinical outcome in breast cancer. Nature 2012;481:389–93.

Zhang, Y. *et al*. (2008) Model-based Analysis of ChIP-Seq (MACS). *Genome Biol.* **9**, R137.