Triform: peak finding in ChIP-Seq enrichment profiles for transcription factors

Karl Kornacker[*]and Tony Håndstad[†]

October 1, 2012

A guide for using the Triform algorithm to predict transcription factor binding sites from ChIP-Seq data

Contents

[*] kornacker@midohio.twcbc.com

[†] tony.handstad@gmail.com

1 Licensing

This package is available under the GPL 2.0. license.

2 Introduction

Chromatin immunoprecipitation combined with high throughput sequencing (ChIP-Seq) is currently the method of choice for genome-wide mapping of binding sites for transcription factors on DNA. An essential step in the analysis of ChIP-Seq data is the genome-wide identification of enriched (peak) regions. As ChIP-seq data can be noisy Park (2009), it can be challenging to identify all significantly enriched regions in a reliable way, and with an acceptable false discovery rate Rye et al. (2011).

The Triform algorithm Kornacker et al. (2012) represents an improved approach for automatic identification of peaks in ChIP-Seq enrichment profiles. The method uses model-free statistics to identify peak-like distributions of sequencing reads, taking advantage of improved peak definition in combination with known characteristics of ChIP-Seq data.

The statistical test in Triform is fully nonparametric, i.e. free from any assumed relationships or fitted parameters. In particular, the test is free from any assumed background model and is therefore more robust than model-based tests, which depend on locally uniform background models and fitted background parameters.

The algorithm identifies triangle-peak-like shapes from the distribution profile of ChIP-Seq reads. A peak region is defined as a region with a significantly negative mean second derivative of the read coverage profile. Such regions have inherently limited width, core sub-regions are directly identified, and these can be used to test for well-defined shifts between overlapping profiles on opposite strands. The test can also handle overlapping peaks. Consequently, the Triform algorithm is able to reject false positive noisy plateaus, thereby increasing specificity with little or no loss of sensitivity.

Like other related algorithms (e.g. FindPeaks Fejes et al. (2008)), Triform tests for local peak-like coverage distributions, but achieves greater sensitivity, specificity and control of FDR by utilizing the Hoel test for detection of significant Poisson inhomogeneities Hoel (1945). Triform computes Hoel test statistics at each position x on each strand, testing whether the reads coverage at x is significantly higher than the average coverage at the two flanking positions $x - d$ and $x + d$ (default d=150bp). The probability distribution of the Hoel test statistic is approximately standard normal for arbitrary nonzero coverage, enabling accurate calculation of p-values which are generally low enough to assure strong control of false discovery rate.

Triform takes advantage of multiple peak profile characteristics. These characteristics include the shift property, which occurs because the full sequence fragments, typically with an average length around 200bp, are sequenced only 25-50bp from each side. The algorithm can use independent control samples, and handles biological replicates.

Triform has been shown to outperform several existing methods in the identification of representative peak profiles in curated benchmark data sets for the transcription factors NRSF/REST, SRF and MAX Rye et al. (2011). In many cases, Triform is able to identify peaks that are more consistent with biological function, compared with other methods. We refer to the paper Kornacker et al. (2012) for further theoretical background on the method, a full description of each step of algorithm, thorough comparisons with other methods, and a small case-study.

3 Overview of Triform

Usage of Triform is split in two steps. In the preprocessing step, information describing ChIP-seq tags in the form of BED-formatted files are converted to a format that describes the tag counts along the chromosomes on the different strands. The BED format is a tab-delimited format where each line describes the position of a mapped read (tag) in the form of space/chromosome, start, end, name, score, strand. Triform will ignore the name and score columns as they are not relevant here. Both control signal (i.e. ChIP-seq reads for control experiments without a TF-specific antibody) and up to several different TF signal files can be processed in the same run. After the preprocessing step, triform itself can be run and will then output the enriched regions.

Both the preprocessing step and the triform step consists of running a single function. Both functions require certain parameters. It is easiest to use a configuration file to supply these parameters, but the parameters can also be supplied directly to the preprocessing or triform function. The configuration file must be in YAML format. See `http://biostat.mc.vanderbilt.edu/wiki/Main/YamlR` for a description of YamlR and `http://cran.r-project.org/web/packages/yaml/index.html` for a description of the yaml R package. An example configuration file is available under the inst/extdata directory in the triform package, and its contents is also shown below.

4 Parameters and configuration file

A total number of 12 parameters must be set to run Triform. These are most easily supplied using a configuration file in the YAML-format. Each line contains the parameter name and value separated by a colon. Some parameters can take multiple values, these values are then given one per line with a dash before the value. Below is an example of a configuration file. The text after the hashes are comments, explaining the purpose of the parameter.

```
READ.PATH : ./tmp  ## Path to source files (reads in BED format)
COVER.PATH : ./chrcovers ## Path for chromosome coverage files
OUTPUT.PATH : ./tmp/Triform_output.csv ## Path for output file (including filename)

TARGETS :
## Filenames for TF experiments
## Must include replicate name (_rep1 or _rep2), and .bed file ending
 - srf_huds_Gm12878_rep1.bed
 - srf_huds_Gm12878_rep2.bed

CONTROLS :
## Filenames for control/background experiments
## Must include replicate name (_rep1 or _rep2), and .bed file ending
 - backgr_huds_Gm12878_rep1.bed
 - backgr_huds_Gm12878_rep2.bed

READ.WIDTH : 100  ## Extended read width (used when preprocessing data) (w)
FLANK.DELTA : 150 ## Fixed spacing between central and flanking locations (d)
MAX.P : 0.1       ## Maximum p-value (used to calculate min.z)

MIN.WIDTH : 10  ## Minimum peak width (min.n)
MIN.QUANT : 0.375 ## Minimum quantile of enrichment ratios.
MIN.SHIFT : 10  ## Minimum inter-strand lag between peak coverage distributions

CHRS : ## Chromosomes to be used in Triform peak detection
 - chrY
```

5 Preprocessing BED files

Start by loading the triform package.

```
> library(triform)
```

This will make the functions "preprocess" and "triform" available. Here, we will use sample data available in the inst/extdata directory for the package. A configuration file similar to the one shown above is also available in this directory. For this vignette, we must get the correct paths to the package at run-time using the system.file function and supply the paths as additional arguments. These will override any settings (i.e. paths) in the configuration file.

```
> config.file.path = system.file("extdata", "config.yml", package="triform")
> data.file.path = system.file("extdata", package="triform")
> preprocess(config.file.path, params=list(READ.PATH=data.file.path, COVER.PATH=data.

/tmp/RtmpknMgnR/Rinst320825e61070/triform/extdata/chrY_backgr_huds_Gm12878_rep1.RData
Loaded /tmp/RtmpknMgnR/Rinst320825e61070/triform/extdata/chrY_backgr_huds_Gm12878_rep
/tmp/RtmpknMgnR/Rinst320825e61070/triform/extdata/chrY_backgr_huds_Gm12878_rep2.RData
Loaded /tmp/RtmpknMgnR/Rinst320825e61070/triform/extdata/chrY_backgr_huds_Gm12878_rep
/tmp/RtmpknMgnR/Rinst320825e61070/triform/extdata/chrY_srf_huds_Gm12878_rep1.RData
Loaded /tmp/RtmpknMgnR/Rinst320825e61070/triform/extdata/chrY_srf_huds_Gm12878_rep1.R
/tmp/RtmpknMgnR/Rinst320825e61070/triform/extdata/chrY_srf_huds_Gm12878_rep2.RData
Loaded /tmp/RtmpknMgnR/Rinst320825e61070/triform/extdata/chrY_srf_huds_Gm12878_rep2.R
```

Each replicate of each TF or control signal should be in its own BED file. The preprocessing will first convert all files with the .bed-extension in the READ.PATH directory to IRanges RangedData objects and save them as RData files. Thereafter, the preprocessing will use the READ.WIDTH parameter to divide each chromosome into segments and calculate for each signal and strand, the number of reads in each segment. The preprocessing ends by saving one file for each chromosome in the dataset, combining all signals and replicate information for the given chromosome in one file.

6 Running Triform

After preprocessing, Triform can be run similarly, by supplying the configuration file path and COVER.PATH to the triform function:

```
>   triform(config.file.path, params=list(COVER.PATH=data.file.path))
```

Triform will then process each chromosome and output each predicted peak region to a file whose path was given in the OUTPUT.PATH parameter.

Note that it is also possible to run preprocessing and Triform by supplying all the parameters directly instead of using a configuration file. In that case, populate a named list with the parameters and consider setting the configPath parameter to NULL. Parameters supplied in the params list will overwrite the values set by any parameters in the configuration file.

```
preprocess(configPath=NULL, params=list(READ.PATH="./inst/extdata",
           COVER.PATH="./inst/extdata", READ.WIDTH=100))

triform(configPath=NULL, params=list(COVER.PATH = "./inst/extdata",
        OUTPUT.PATH = "./inst/extdata/Triform_output.csv",
        MAX.P = 0.1, MIN.WIDTH = 10, MIN.QUANT = 0.375, MIN.SHIFT = 10,
        FLANK.DELTA = 150, CHRS = c("chrY"), CONTROLS =
        c("backgr_huds_Gm12878_rep1.bed", "backgr_huds_Gm12878_rep2.bed"),
        TARGETS=c("srf_huds_Gm12878_rep1.bed", "srf_huds_Gm12878_rep2.bed")))
```

7 Session info

```
>   sessionInfo()
```

```
R version 2.15.1 (2012-06-22)
Platform: x86_64-unknown-linux-gnu (64-bit)

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_US.UTF-8        LC_COLLATE=C
 [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=C                 LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base
```

```
other attached packages:
[1] triform_1.0.0      yaml_2.1.4         IRanges_1.16.0     BiocGenerics_0.4.0

loaded via a namespace (and not attached):
[1] parallel_2.15.1 stats4_2.15.1   tools_2.15.1
```

References

A.P. Fejes, G. Robertson, M. Bilenky, R. Varhol, M. Bainbridge, and S.J. Jones. FindPeaks 3.1: a tool for identifying areas of enrichment from massively parallel short-read sequencing technology. *Bioinformatics*, 24 (15):1729–1730, 2008.

P.G. Hoel. Testing the homogeneity of poisson frequencies. *The Annals of Mathematical Statistics*, 16(4):362–368, 1945.

K. Kornacker, M.B. Rye, Håndstad T., and F. Drabløs. The Triform algorithm: improved sensitivity and specificity in chip-seq peak finding. *BMC Bioinformatics*, 2012. In press.

Peter J. Park. ChIP-seq: advantages and challenges of a maturing technology. *Nat Rev Genet*, 10(10):669–680, October 2009. ISSN 1471-0056.

M.B. Rye, P. Sætrom, and F. Drabløs. A manually curated chip-seq benchmark demonstrates room for improvement in current peak-finder programs. *Nucleic Acids Research*, 39(4):e25, 2011.