**Methods for the analysis of Topologically Associating Domains (TADs)**

i. Running Head

ii. Summary/Abstract

In one or two paragraphs, please write a brief overview of the method described (this is the abstract that appears on sites such as PubMed).

iii. Key WordsPlease include 5-10 key words for referencing by electronic databases and search engines.

1. **Introduction**

*This section should contain a summary of, and the outline of any theory to, the method that you’re are describing. It should also outline the major procedures involved in the protocol.*

1. **Materials**

*This section should list the composition of all buffers, media, solutions, and specialist equipment etc., that are necessary for carrying out the method described in section 3. Suppliers aren’t needed for routine reagents (the reader will use his/her own local supplier) and catalogue numbers aren’t required at all for reagents. All buffers, solutions and media should be presented in the same format i.e. name, colon then composition on one continuous line, with components separated by commas not semi-colons. Do not list individual reagents, but do describe the composition of the solutions, etc they are used in.*

*Materials used for analysis of Hi-C data primarily include computers running Unix operating system and specific software to perform data processing, TAD and interaction calling, and visualization and annotation of results. The software used in this chapter is listed in Subsection 2.1.2.*

* 1. **Necessary resources**
     1. *Hardware*

UNIX operating systems with AWK package installed, standard CPUs or GPUs.

* + 1. *Software*

The following software and tools are required to analyze the example data:

* R version 3.6.0 (R Core Team 2019)
* TopDom R package (Version 0.8.1; see Note 1): <https://github.com/HenrikBengtsson/TopDom>
* CaTCH R package (Version 1.0):

<https://github.com/zhanyinx/CaTCH_R>

* pre (Juicebox and Juicer Tools Version 1.5.3; Durand et al. 2016): <https://github.com/aidenlab/juicer/wiki/Pre>
* Arrowhead (Juicebox and Juicer Tools Version 1.5.3; Durand et al. 2016): <https://github.com/aidenlab/juicer/wiki/Arrowhead>
* R implementation of the Measure of Concordance:

<https://github.com/CSOgroup/TAD-benchmarking-scripts/blob/master/Figure2/fig2_fig3_fig4_fig5_moc_calc.R>

* supp. R code with helper functions (e.g. file conversion) → GitHub ?

For installation refer to the help page of each tool.

* 1. **Datasets**
     1. *Description of the example data*
     2. *Download of the data*

1. **Methods**

*This is the main section and should explain in detail the individual steps necessary to carry out the technique. Where possible, please simply list the steps in numerical order. For techniquesthat comprise a number of separate major procedures, please indicate these separate procedures in the introduction, and then subdivide section 3 into subheadings to cover each procedure (3.1, 3.2 etc). The steps in each subsection should then be numbered individually, renumbering from number one. Do take great care to try to indicate any little "tricks" or nuances that help improve your method by referring to relevant "notes" in section 4 (see below). This sort of information rarely gets into the scientific literature. You may also find it useful to relate to some aspects of the theory in this section indicating the purpose of some of the major steps by cross-referencing to an appropriate “note”. Do not be tempted to get involved in the description of variations/alternatives to your technique in this section: this can be done in the "Notes" section. Stick to the basic procedure detailed in this section.This section must be comprehensive. Do not send the reader away to find information for a particular step in another reference. All relevant practical detail must be given in this section.*

For the ease of the analyses and downstream post-processing, all softwares were called directly from an R terminal.

* 1. **TAD callling**
     1. *Detection of TADs with TopDom*

*Algorithm outline*

TopDom (Shin et al. 2016) proceeds in three steps. First, it computes a binSignal value that represents, for each bin, an average of the counts between upstream and downstream regions around it. This metric is expected to be low at TAD boundaries and high in mid-position of the domains. Then, the second step consists in detecting minima (i.e. candidate TAD boundaries) in the curve formed by the series of binSignal values along the chromosome using a piecewise linear function. In the final step of the algorithm, the list of boundaries are refined by retaining those that represent regions with significantly depleted interactions between upstream and downstream intervals, compared to contact counts within these intervals, as assessed with a Wilcoxon rank sum test.

*Input format*

The expected format is a N x (N+3) matrix of normalized counts. The first three columns hold information about the chromosome, start and end of the bin (genomic coordinates). The remaining matrix entries contain the contact values. To be passed to TopDom, the matrix can be stored in a text file or gzipped format. A N x (N+4) format, with an additional id-column, is also accepted.

Chr10 0 40000 0 0 0 0 ...

chr10 40000 80000 0 0 0 0 ...

chr10 80000 120000 0 0 0 0 ...

chr10 120000 160000 0 0 0 0 ...

*Parameters*

The unique free parameter required by TopDom is a window size, that controls the width of the region considered around each bin for calculating the binSignal at the first step, and for the filtering of false positives at the final step. In the original article, the authors recommend using a value between 5 and 20, and identify 5 as the most appropriate in their analysis.

*Launching*

Once the package installed and loaded in R, launching of TopDom is simply performed using the TopDom function on each chromosome (cf. supp. R code).

topDom\_out <- TopDom(topDom\_file, window.size=5, outFile=topDom\_outFile)

topDom\_tads\_dt <- topDom\_out[["bed"]]

*Output format*

The output of the TopDom function consists in a list of 3 data frames (“bed”, “binSignal” and “domain”). These 3 data frames will automatically be written in the “<topDom\_outFile>.bed”, “<topDom\_outFile>.binSignal” and “<topDom\_outFile>.domain” text files respectively. To retrieve a BED-like format list of TADs, the most straightforward way is to extract the “bed” table of this list and, using the “name” column, to filter the rows containing the “domain“ coordinates (this table also holds information about gaps; cf. supp. R code). Thus, the resulting genome partition will also contain inter-TAD regions. The domains are non-overlapping.

* + 1. *Detection of TADs with CaTCH*

*Algorithm outline*

CaTCH (Zhan et al. 2017) is an algorithm that uses a metric called reciprocal insulation (RI) to quantify how well a TAD is isolated from its neighbors. For example, a RI of 0.5 between two adjacent domains corresponds to average interactions between them are 50% smaller than those within them. Hence, a large RI value indicates that the two domains are separated by a strong boundary. The method starts by segmenting the Hi-C map into a set of domains, and then merging consecutive TADs whose RI score is lower than a given threshold. In order to stratify the whole hierarchy of TADs, CaTCH systematically varies this latter value.

*Input format*

The expect input format for CaTCH is a data frame in 4-column format, where each rows contains the normalized count value (4th column) between two given bins (2nd and 3rd column; integer values). The user guide of CaTCH does not specify if the bin numbering should be 0- or 1-based. Text files as well as gzipped files are accepted.

chr10 1001 1001 540.6538

chr10 1001 1002 246.8385

chr10 1001 1003 130.4811

chr10 1001 1004 126.1130

*Parameters*

By default, CaTCH identifies domains for a range of insulation thresholds (from 0.001 to 1 by 0.001), and doing so detects increasingly larger, more strongly insulated domains. After completion of the domain calling, a set of TADs can be manually extracted for any given threshold (smaller value yields to higher number of (smaller) domains). In the original publication, the authors report that a value of 0.65 leads to maximal CTCF enrichment at boundaries, and the best match with the directionality index method (Dixon et al. 2012) is obtained with 0.69.

*Launching*

Once the package installed and loaded in R, launching of CaTCH is simply performed using the domain.call function. If TAD calling is performed for multiple chromosomes, multi-threading can be achieved by calling instead the domain.call.parallel function.

CaTCH\_out <- domain.call(catch\_file)

CaTCH\_dt <- CaTCH\_out[["clusters"]]

*Output format*

The output of CaTCH consists in a list of two data frames. One (“ncluster”) stores the number of domains identified for each of the insulation thresholds; the other table (“clusters”) contains information about the insulation and location of the domains for all the threshold values. A list of TADs in BED-like format can easily be derived from this latter table by extracting the rows at the desired insulation threshold, and converting back bin number (1-based or 0-based depending on the input format) to genomic coordinates (cf. supp. R code). With this algorithm, the genome is fully partitioned into non-overlapping domains (no gaps).

* + 1. *Detection of TADs with Arrowhead*

*Algorithm outline*

Arrowhead (Rao et al. 2014) starts with a transformation of the input count matrix (M) into an arrowhead matrix A as follows: Ai,i+d= (Mi,i-d-Mi,i+d)/(Mi,i-d +Mi,i+d). Thus, the value of Ai,i+d a reflects the preference of directionality of the locus i for the contacts inside an interval distance of size d. As such, it will take a large positive value if the locus *i-d* is inside the domain, but not *i+d*. On the opposite, when *i+d* lies inside and *i-d* outside, Ai,i+d will be strongly negative. And it will be almost zero when both loci are outside or inside the domain. Geometrically, if there is a TAD between *a* and *b*, it will be apparent in M as a pair of “lower” and “upper” triangles, as these operations sharpen the contours of potential domains. More precisely, if there is a domain corner at the position (*a*,*b*), an “upper” (“intra-domain”) triangle (U) of strongly negative entries in M with coordinates (*a*,*a*), (*a*,*b*), and ((*a+b*)/2,*b*) as well as a “lower” (“out-of-domain”) triangle (L) of large positive values with vertices at ((*a*+*b*)/2,*b*), (*b*,*b*), and (*b*,*2b-a*) will be apparent. Therefore, this pattern can be quantified with a “corner score” that combines: i) the sum of the signs in L minus sum of the signs in U; ii) the sum of the values in L minus sum of the values in U; iii) total variance in U and L. The heuristic for calling domains consists then in detecting their corners by finding cells of M with high corner scores by applying an empirical threshold to the normalized subscores.

*Input format*

The arrowhead program requires files in the specific “hic” format as input, (“a highly compressed binary file that provides rapid random access to the matrices”). Appropriately, the Juicer pipeline provides another Java tool (pre) that can be used to convert count data from text files to hic files (several input formats can be handled; cf. supp. R code).

*Parameters*

To run Arrowhead, the user should set the size of the sliding window along the diagonal in which TADs will be detected (-m command line flag). This should be an even number and its default value is 2000. Arrowhead can work on already normalized count data or perform the normalization internally (controlled via the -k command line flag).

*Launching*

Arrowhead can be launched from a terminal by calling Java with the appropriate command. The command can directly specify on which chromosome(s) Arrowhead will be run (-c flag), and the --threads flag can be used to achieve multi-threading.

*Output format*

After completion of the program, an output file (“<resolution>\_blocks”) containing information about the domains found and some statistics will be written. In this latter, each row stores information for a domain, with its start and end genomic coordinates (not sorted). The partition obtained from Arrowhead can contain gaps. In addition, domains can be nested or overlapping. Therefore, some additional post-processing might be necessary, depending of the desired output (cf. supp. R code).

* 1. **Comparison of genome partitions**
     1. *The Measure of Concordance*

The Measure of Concordance (MoC), previously introduced to compare clustering partitions (Pfitzner et al. 2009), can be used to assess the similarity between two TAD caller results. Briefly, given two sets of TADs, MoC assesses the overlap between each pair of TADs, measured in number of base pairs and considering the overall size of both TADs. MoC ranges from 0, complete lack of concordance, to 1, perfect concordance, and it has the desirable property of being symmetric.

Using the publicly available implementation of the measure of concordance, the two partitions to be compared can simply be passed as 3-column BED format (chromosome, start, end) to the get\_MoC function, either by providing the path to corresponding BED file or as a data frame R object. Some additional parameters can be used to control e.g. how to handle comparison with a gap and a domain or if a penalty for the number of TADs should be introduced (cf. description in the source code). To speed up the calculation, multi-threading is supported (nCpu parameter).

# calling the function on the text files:

get\_MoC("out\_Arrowhead/arrowhead\_final\_domains.txt", "out\_CaTCH/CaTCH\_final\_domains.txt") # 0.55

get\_MoC("out\_Arrowhead/arrowhead\_final\_domains.txt", "out\_TopDom/topDom\_final\_domains.txt") # 0.54

get\_MoC("out\_CaTCH/CaTCH\_final\_domains.txt", "out\_TopDom/topDom\_final\_domains.txt") # 0.65

# calling the function on the R data frame objects:

topdom\_dt <- read.delim("out\_TopDom/topDom\_final\_domains.txt", header=FALSE, col.names=c("chromo", "start", "end"))

catch\_dt <- read.delim("out\_CaTCH/CaTCH\_final\_domains.txt", header=FALSE, col.names=c("chromo", "start", "end"))

arrowhead\_dt <- read.delim("out\_Arrowhead/arrowhead\_final\_domains.txt", header=FALSE, col.names=c("chromo", "start", "end"))

get\_MoC(topdom\_dt, catch\_dt) # 0.65

get\_MoC(topdom\_dt, arrowhead\_dt) # 0.54

get\_MoC(arrowhead\_dt, catch\_dt) # 0.55

1. **Notes**

*As we all know, even the simplest techniques go wrong from time to time. Would you therefore indicate any major problems or faults that can occur with your technique? Try to indicate themajor sources of problems and how they can be identified and overcome. With reference torelated techniques, any variations of the technique that you have described should also be made in this section, as well as--where relevant--an indication of the sensitivity of the method, timescale for the singled technique, etc. This "Notes" section is a hallmark of this series and has been singled out for praise by a number of reviewers. Please try and make this section as extensive aspossible by putting on paper all of your various experiences with the technique. Each ‘Note’should be cross-referenced with the ‘Materials’ and ‘Methods’ sections, e.g. (see Note 1).*

1. TopDom was first distributed through the Zhou’s lab website (http://zhoulab.usc.edu/TopDom). However, this latter is no longer maintained. Fortunately, an R package has been developed upon of it (Bengtsson and Shin 2018; the original script has been made available on the GitHub account hosting this package). Instructions for the installation of the package are available on the GitHub page (https://github.com/HenrikBengtsson/TopDom).
2. The authors recommend running Arrowhead (and HiCCUPS) only if the matrix contains a large number of contacts (e.g., more than 300 million contacts; sparsity check). If the data do not pass the sparsity check, the algorithm exits with a warning. This control can be overridden adding the --ignore\_sparsity flag to the command line.
3. **References**

Bengtsson, H. and Shin, H. (2018). R Package TopDom: An Efficient and Deterministic Method for Identifying Topological Domains in Genomes. R package version 0.8.1. <https://github.com/HenrikBengtsson/TopDom>

Dixon J. R., et al. (2012). Topological domains in mammalian genomes identified by analysis of chromatin interactions. Nature. 485(7398):376–80.

Durand, N. C. et al. (2016). Juicer provides a one-click system for analyzing loop-resolution Hi-C experiments. Cell Systems. 3(1):95-98.

Pfitzner D. et al. (2009). Characterization and evaluation of similarity measures for pairs of clusterings. Knowledge and Information Systems. 19(3):361.

R Core Team (2019). R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing. Vienna, Austria. [https://www.R-project.org](https://www.R-project.org/).

Rao et al. (2014). A 3D map of the human genome at kilobase resolution reveals principles of chromatin looping. Cell. 159(7):1665-1680

Shin, H. et al. (2016). TopDom: an efficient and deterministic method for identifying topological domains in genomes. Nucleic acids research. 44(7):e70.

Zhan, Y. et al. (2017). Reciprocal insulation analysis of Hi-C data shows that TADs represent a functionally but not structurally privileged scale in the hierarchical folding of chromosomes. Genome research. 7;27(3):479–90.