*Ontoscope:* Determining “identity-defining” transcription factors for various cell types

Vignette authors: Kartikay Chadha, Shivani Kamdar, Burton Mendonca, Anam Qudrat

University of Toronto

Department of Biochemistry

Compiled: April 27, 2016

**Contents**

**1 Introduction 2**

**2 Processing overview 2**

**3 Submodules and workflow**  **z**

3.1 Normalization z

3.2 FANTOM expression data import z

3.3 Background derivation from cell ontology z

3.4 Protein and transcription factor network analysis z

3.4.1 STRING

3.4.2 TRRUST

3.4.3 Regnet

3.5 Differential gene expression analysis z

3.6 Integration of network- and expression-based influence scores z

3.7 Selection of transcription factors needed for conversion z

**4 Post-processing and analysis z**

4.1 Visualization z

4.2 Literature-based confidence score z

4.3 Validation from published experimental data z

**5 Acknowledgments z**

**6 Session info z**

**7 References z**

**1 Introduction**

This document offers an overview of the R package *Ontoscope*, which uses publicly available expression and protein-protein interaction data to computationally determine a list of transcription factors able to facilitate conversion from one cell type to another. Following determination of this transcription factor list, it is also designed to assign confidence scores to each factor based on literature search and validation against published experimental data and/or other computational prediction software.

The *Ontoscope* package is designed to independently validate the findings of Owen Rackham and colleagues’ *Mogrify* ([www.mogrify.net](http://www.mogrify.net))1. Its workflow is based on the published *Mogrify* protocol, with modifications including the usage of updated transcription factor definitions and the inclusion of different regulatory networks (specifically, the TRRUST2 and Regnet3 protein-protein interaction datasets).

This vignette contains an overview of the package workflow, defining initialization parameters and walking through each submodule with examples and usage instructions provided.

**2 Processing overview**

*Ontoscope* determines transcription factors required for conversion based on calculation of network- and expression-based influence scores for differentially expressed factors in the target cell line as compared to the source. Target and source cells are identified by FANTOM consortium cell line IDs, and are restricted to those cell types for which FANTOM gene expression data from CAGE-seq is available.

Calculation involves six primary phases:

1. *Import of expression data:* The first step is to define source and target cell lines, reading in raw FANTOM expression count data for each cell line. Transcription factors are defined from published literature and databases, and all gene names are normalized to HGNC IDs in order to ensure accuracy of comparison between cell lines and datasets. Download of FANTOM count data is automatic upon input of cell line IDs, and does not require further input from the user.
2. *Calculation of background:* Something
3. *Differential expression-based influence score assignment:* Something
4. *Interaction-based influence score assignment:* Something
5. *Integration and binning:* Following generation of differential expression and network influence scores for transcription factors in the source and target cells, the transcription factors are ranked based on a combination of the scores. The ranked transcription factor lists are refined further to produce a characteristic set of transcription factors for a specific cell conversion. This refinement is achieved by: removing transcription factors from the target cell that are already “highly” expressed in the source cell; removing redundant transcription factors with “high” regulatory overlap.
6. *Visualization and validation:* Something

**3 Submodules and workflow**

This section outlines examples and functions underlying the usage of *Ontoscope* to generate lists of transcription factors for conversion.

Users should begin by defining cells of origin and desired target cell type as follows:

> sourcecell <- “eye”

> target <- “fibroblast”

Here, a retinal to fibroblast cell conversion has been used as an example. It is important to define both keywords and FANTOM IDs for desired cell types, as while FANTOM IDs will be used for the actual Ontoscope workflow, keyword definitions are important for visualization and literature validation.

Next, users should select their desired FANTOM IDs for source and target cells by sourcing and running the **fantom\_import** submodule for both source and target cell types. Output for the source cell search is as follows:

> source(“./fantom\_import/fantom\_main.R”)

> fantomSearch(sourcecell)

V1 FANTOM.5.Ontology.ID FANTOM.5.Access.Number

1375 eye - muscle inferior rectus, donor1 FF:10272-104E2 1381

1376 eye - muscle lateral, donor2 FF:10298-104H1 1382

1377 eye - muscle medial, donor2 FF:10299-104H2 1383

1378 eye - muscle superior, donor2 FF:10297-104G9 1384

1379 eye - vitreous humor, donor1 FF:10268-104D7 1385

1380 eye, fetal, donor1 FF:10054-101G9 1386

From the list, the desired specific subtype (ie: lateral eye muscle, vitreous humor, fetal eye) of cell can be selected, and the FANTOM Ontology ID (FF:ID) noted. Here, we have chosen to convert inferior rectus eye muscle to cardiac fibroblast. FF:IDs may then be defined as follows:

> sourceFF <- “FF:10272-104E2”

> targetFF <- “FF:11268-116G8”

Following this, simply running runOntoscope.R will process the entire workflow for you, outputting a list of top transcription factors required for conversion with confidence rankings based on literature as well as heatmap and interaction network visualizations. The runOntoscope.R package also contains built-in functionality to cross-validate the transcription factor lists generated by Ontoscope against published conversion lists from either literature or the MOGRIFY package.

**3.1 Normalization**

Prior to beginning the workflow proper, it is important to first normalize all gene identifiers in order to allow comparison between gene lists and data from different sources. Ontoscope takes as input four main sources of data: gene expression data from FANTOM5, and protein-protein interaction and regulatory network data from STRING4, TRRUST, and REGNET. While FANTOM5 and TRRUST datasets contain HGNC identifiers for genes, STRING primarily identifies genes based on Ensembl IDs, and must be normalized to HGNC symbols for comparison purposes. Furthermore, while REGNET uses HGNC identifiers for its gene interactions, these IDs are four years old at the time of publication of this vignette, and may be outdated.

In order to normalize STRING interactions, the **normalizeWeave.R** submodule is sourced, using the R Bioconductor package biomaRt to create a new datafile from the base STRING data table with approximately 8.5 million protein-protein interactions. In order to avoid extremely lengthy processing times, STRING Ensembl protein IDs are placed in a new data frame as row names, following which a new vector of HGNC symbols is created with biomaRt. The updated file, **curatedOutput.Rdata**, is then available for usage in downstream applications.

Should Regnet files require updated HGNC symbol assignments, Entrez IDs can be retrieved from Regnet files and transcribed into HGNC symbols as well by altering the following lines in **normalizeRegnet.R** :

> IDmap <- data.frame(entrezgene=IDmap, HGNC=””, stringsAsFactors=FALSE)

> BMmap <- getBM(filters = “entrezgene”,

attributes = c(“entrezgene”, output),

values = IDmap$entrezgene,

mart = ensembl)

> colnames(BMmap) <- c(“entrezgene”, output)

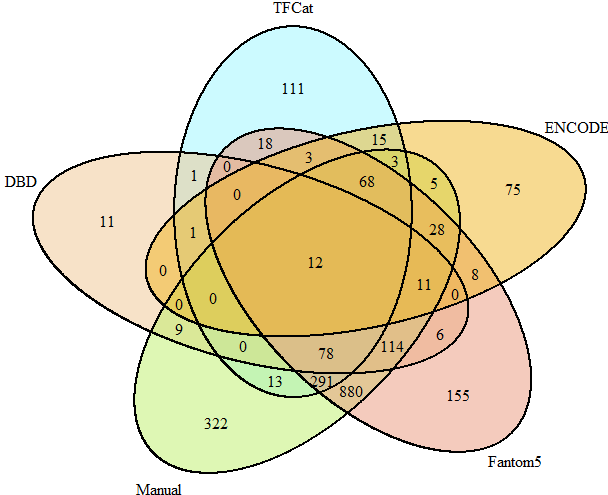
All input datafiles will now have been normalized to identify genes by HGNC symbols.

Finally, transcription factors must be defined prior to performing analysis. Sourcing the **normalizeTF.R** subscript will automatically compile a list of transcription factors common among at least two out of the five following datasets:

1. The TFCat database
2. The DBD (DNA-binding domain) database
3. A manually assembled list of transcription factors with annotations5
4. The FANTOM5 transcription factor dataset
5. ENCODE transcription factor data derived from their ChIP-seq antibody list

The choice of how many lists to use as a cutoff was based on variations in overlap between the lists **(Figure 1)**, and may be easily modified through subsetting of the MergedList table generated by this submodule, which lists all transcription factors and whether or not they are present in each dataset, via modification of the following line:

> TFList <- unique(MergedList[!MergedList$Count==”1”,])$Gene.Symbol



**Figure 1: Varying degrees of overlap between human transcription factor datasets.** Venn diagram depicting large overlap between the manually curated and Fantom5 lists, while other datasets possess less unique and overlapping transcription factors. Generated by the VennDiagrams package: draw.quintuple.venn()

Note that all transcription factors have been normalized to HGNC symbols where appropriate. The normalize submodule allows for easy replacement or update of lists through the replacement of their .csv datafiles with updated data.

**3.2 FANTOM expression data import**

Expression profiles for genes across available cell lines have been derived from cap analysis of gene expression (CAGE) sequencing by the FANTOM5 consortium. The FANTOM import submodule allows for the import and processing of raw read counts from this sequencing data, accepting several different possible search terms as inputs and allowing for import of normalized read counts as an alternative. For the purposes of the Ontoscope workflow, the **fantom\_main.R** submodule has already been sourced. Furthermore, as we perform our own normalization and background derivations further downstream, we must focus on importing raw read counts rather than normalized expression data.

The **runOntoscope.R** script first searches the FANTOM5 database for matching FF IDs and downloads the raw expression counts for each as follows:

> FFVect <- c(sourceFF, targetFF)

> fantomOntology(FFVect)

[1] "Sample\_DB Loaded!"

Returning RAW COUNTS

MATCHED: 2 of 2

2 Search Result(s) Were Found. Loading...

Loading Results from Fantom Access Number 1381 ( 1 / 2 ) ...

Results from Fantom Access Number 1381 Loaded!

Loading Results from Fantom Access Number 377 ( 2 / 2 ) ...

Results from Fantom Access Number 377 Loaded!

All results have been loaded into fantomResults

fantomResults is a large, subsettable list of dataframes containing genetic annotations, peak numbers, gene names, and HGNC, Uniprot, and Entrez Gene IDs for every sample, with one dataframe per sample. These results can be summarized in one dataframe, with normalized HGNC symbol gene names and raw expression counts for each gene, through the fantomSummarize command:

> fantomSummarize(5)

Preparing the Genes

Summarizing:eye - muscle inferior rectus, donor1.CNhs13444.10272-104E2

Summarizing:Fibroblast - Cardiac, donor1.CNhs12498.11268-116G8

Filtering Relevant Results. This step takes awhile ...

Preparing Normalized Gene Names ...

All Genes Normalized!

Fixing Duplicates ...

Applying Threshold ...

Your results have been summarized in: fantomCounts!

The bracketed number indicates the minimum threshold for read counts in each gene. Genes with read counts less than the threshold number (here, 5) will not be included in the summarized fantomCounts

file, which is now ready for downstream processing.

**3.3 Background derivation from cell ontology**

**3.4 Protein and transcription factor network analysis**

3.4.1 STRING

3.4.2 TRRUST

3.4.3 Regnet

**3.5 Differential gene expression analysis**

**3.6 Integration of network- and expression-based influence scores**

**3.7 Selection of transcription factors needed for conversion**

Transcription factors required for a specific cell conversion are predicted using a combination of the differential expression and network influence scores. The **OntoscopeRank.R** script includes functions to predict a characteristic set of TF for a specific cell conversion.

The main inputs required to run this script are dataframes containing gene scores for both the source and target cell in a cell conversion. The first column in each dataframe is a gene ID, with gene scores in subsequent columns. Examples of two gene score data frames (differential expression and network influence) are shown below. The gene score dataframes must be filtered to only included transcription factors, using the getTFs function, which is described in the table of helper functions below.

|  |  |
| --- | --- |
| **Differential expression scores example (diffScores)** | **Network influence scores example**  **(netScores)** |
| gene gsx  1 gene 14 0.01040496  2 gene 12 0.70933139  3 gene 10 1.56673569  4 gene 20 -0.44271365  5 gene 15 -0.24856313  6 gene 17 -0.36300387  7 gene 19 -0.33784355  8 gene 1 2.06647221  9 gene 18 1.18247259  10 gene 8 -0.30116691 | gene tisSTRING tisTRRUST  1 gene 16 0.2777882 -0.7247869  2 gene 8 -0.5534669 0.1473134  3 gene 11 -0.6729669 -1.6007885  4 gene 7 2.4737518 1.0816131  5 gene 15 -0.3292432 1.2367742  6 gene 17 -0.2664347 0.4147440  7 gene 6 0.1938819 -0.3247035  8 gene 20 -0.2319178 -1.3374891  9 gene 9 0.6298599 -0.5337684  10 gene 13 -0.1629077 0.2811862 |

The rankGeneDataFrame function takes as inputs filtered gene score data frames, as separate arguments, as well as the length of rankings. For example, the command below ranks each gene based on the example gene score data frames above. The rankGeneNumber argument specifies the length of the ranking for each score. In this case, Top 5 rankings are calculated.

rankedDataFrame <- rankGeneDataFrame(diffScores, netScores, rankGeneNumber = 5)

The rankGeneDataFrame function ranks each gene by the different scores, limiting the rankings the value specified by the user (e.g. Top 100, Top 10). If a gene does not belong to the ranking for a particular score, it is given the last rank (e.g. if the ranking is for the Top 100 differentially expressed genes, genes that are outside the Top 100 are all assigned a rank of 100). Finally, it sums the ranks for each score to obtain a final rank for each gene.

The output dataframe of the rankGeneDataFrame function contains the genes sorted by their final rank. The first column in each dataframe is a gene ID. The second column contains the final ranks, with subsequent columns containing ranks based on the different gene scores.

The output produced for the example above is shown below:

gene finalRank gsxRank tisSTRINGRank tisMARARank

1 gene 7 8 5 1 2

2 gene 1 11 1 5 5

3 gene 15 11 5 5 1

4 gene 10 12 2 5 5

5 gene 9 12 5 2 5

6 gene 18 13 3 5 5

7 gene 16 13 5 3 5

8 gene 17 13 5 5 3

9 gene 12 14 4 5 5

10 gene 6 14 5 4 5

11 gene 13 14 5 5 4

12 gene 14 15 5 5 5

13 gene 8 15 5 5 5

The helper functions for the rankGeneDataFrame function are the following:

| **Function** | **Description** |
| --- | --- |
| rankGenes <- function(geneScoreDataFrame, geneNumber = 100) | Ranks a 2-column geneScoreDataFrame, whose first column is gene IDs and whose second column is gene scores. No. 1 rank means highest score. Ranks are limited by geneNumber. |
| getTFs <- function(geneDataFrame, TFList) | Filters a geneDataFrame to only contain transcription factors belonging to TFList. |

After ranking the transcription factors in the source and target cell types, these two ranked dataframes are compared using the compareTFDataFrame function, in order to remove transcription factors from the target cell type dataframe that are already “highly” expressed in the source cell type. The purpose of this function is to create a more refined “cell reprogramming” set of transcription factors for the cell conversion.

The inputs for this function are: the rankGeneDataFrame function outputs for the source and target cell; a data frame containing source cell expression data with gene ID in the first column and expression level in the second column; an expression threshold, above which expression is considered “high”. An example of the source expression dataframe is shown below:

gene expression

1 gene 10 46.00291

2 gene 3 30.98600

3 gene 8 74.52001

4 gene 17 30.07706

5 gene 20 56.65336

6 gene 12 49.66658

7 gene 16 34.63197

8 gene 1 34.83485

9 gene 14 19.99675

10 gene 15 37.59648

11 gene 2 68.15543

12 gene 5 64.49444

13 gene 4 10.90653

14 gene 6 41.19423

15 gene 19 63.94464

16 gene 9 47.26947

17 gene 7 38.71289

18 gene 13 55.29177

19 gene 18 53.40839

20 gene 11 46.49910

The compareTFDataFrame function can be called as follows:

comparedDataFrame <- compareTFDataFrame(sourceRankedTFs, targetRankedTFs, sourceExpression, expressionThreshold = 20)

The compareTFDataFrame function goes through each TF in the target cell dataframe. If that TF is present in the source cell dataframe and is expressed at a level above the expression threshold, according to the source expression dataframe, then that TF is removed from the.

The desired output of the compareTFDataFrame function is a “cell reprogramming” set of transcription factors. An example of the output is shown below. Note, the format is the same as the rankGeneDataFrame output.

gene finalRank gsxRank tisSTRINGRank tisMARARank

1 gene 14 8 2 1 5

4 gene 4 11 1 5 5

5 gene 20 12 5 2 5

6 gene 19 13 5 3 5

10 gene 3 15 5 5 5

Finally, the “cell reprogramming” transcription factor dataframe is pruned to remove redundant TFs that have “high” regulatory overlap. This pruning is achieved via the pruneTFDataFrame function.

The inputs to the pruneTFDataFrame function are: the output dataframe of the compareTFDataFrame function; a regulatory network, a directed igraph network with genes as nodes and edges meaning “regulates”, produced in the **WEAVE.R** script; a coverage similarity threshold, above which two transcription factors regulatory neighborhoods are considered “highly” overlapped.

The pruneTFDataFrame function can be called as follows:

prunedDataFrame <- pruneTFDataFrame(comparedDataFrame, regNetwork, coverageSimilarity = 0.98)

The function goes through each transcription factor in the dataframe. Next, it compares the genes regulated by that transcription factor with the regulated genes of every higher ranked transcription factor. If a higher ranked transcription factor contains a certain percentage of the regulated genes of the lower ranked transcription factor, the lower ranked transcription factor is removed, or “pruned”, from the dataframe.

The pruneTFDataFrame output is a dataframe containing a minimal set of transcription factors required for a specific cell conversion, with little regulatory overlap. An example of the output is shown below. Again, the format of the output is the same as those of rankGeneDataFrame and compareTFDataFrame.

gene finalRank gsxRank tisSTRINGRank tisMARARank

1 gene 14 8 2 1 5

4 gene 4 11 1 5 5

10 gene 3 15 5 5 5

**4 Post-processing and analysis**

**4.1 Visualization**

**4.2 Literature-based confidence score**

**4.3 Validation from published experimental data**

**5 Acknowledgments**

This package was developed as part of the BCB420/JTB2020 bioinformatics class as part of the Department of Biochemistry at the University of Toronto, Canada, with the support of Dr. Boris Steipe. We wish to acknowledge all co-authors of *Ontoscope* package modules, who have equally contributed to the development of this project: Julian Mazzitelli, Dmitry Horodetsky, Christopher Li, Kartikay Chadha, Fupan Yao, Shivani Kamdar, Dan Litovitz, Howard Wu, Anam Qudrat, Jialun Tom Chen, Ryoga Li, Eugenia Barkova, Burton Mendonca, Phil Fradkin, and Boris Steipe.

**6 Session Info**

> sessionInfo(package=NULL)

(The output of the above command needs to be filled in by Dr. Steipe after integrating all the code and running it).

**7 References**

[1] Rackham OJ, Firas J, Fang H, Oates ME, Holmes ML, Knaupp AS, FANTOM Consortium, Suzuki H, Nefzeger CM, Daub CO, Shin JW, Petretto E, Forrest AR, Hayashizaki Y, Polo JM, Gough J. A predictive computational framework for direct reprogramming between human cell types. *Nat Genet,* 48(3): 331-335, 2016.

[2] Han H, Shim H, Shin D, Shim JE, Ko Y, Shin J, Kim H, Cho A, Kim E, Lee T, Kim H, Kim K, Yang S, Bae D, Yun A, Kim S, Kim CY, Cho HJ, Kang B, Shin S, Lee I. TRRUST: a reference database of human transcriptional regulatory interactions. *Sci Rep,* 5:11432, 2015.

[3] Chi SM, Seo YK, Park YK, Yoon S, Park CY, Kim YS, Kim SY, Nam D. REGNET: mining context-specific human transcription networks using composite genomic information. *BMC Genomics,* 15:450, 2014.

[4] Szklarczyk D, Franceschini A, Wyder S, Forslund K, Heller D, Huerta-Cepas J, Simonovic M, Roth A, Santos A, Tsafou KP, Kuhn M, Bork P, Jensen LJ, von Mering C. STRING v10: protein-protein interaction networks, integrated over the tree of life. *Nucleic Acids Res*, 43:D447-52, 2015.

[5] Vaquerizas JM, Kummerfield SK, Teichmann SA, Luscombe NM. A census of human transcription factors: function, expression and evolution. *Nat Rev Genet*, 10(4):252-263, 2009.