GAZE

Release 0.0.1

Fatemeh Behjati

Feb 10, 2022

CONTENTS:

GAZE is a comprehensive tool to integrate multi-modal high throughput data with single-cell RNA-seq measurements to build a statistical model that is able to infer transcription factor (TF) binding activity at single cell resolution. To achieve this, the user requires to provide the input data either directory to GAZE or using encapsulated form of inputs generated by the STARE package.

CHAPTER

ONE

INDICES AND TABLES

- genindex
- modindex
- search

CHAPTER

TWO

GETTING STARTED WITH GAZE

2.1 Installation

Under construction

2.2 Dependencies

Under construction

CHAPTER

THREE

INPUT ARGUMENTS

- User can provide their own feature data, which means it doesn't have to be obtained via STARE
- **When it should be created via STARE, here is the list of input arguments:**
  - **bed file containing the regions (chr, start, end).**
    - with only the above mentioned three columns a static mode of feature will be created
    - if in addition to the reagion info one activity column is provided (bulk DNase-seq or ATAC-seq), then the epigenetic mode will be executed
    - if multiple columns are provided after the region columns, then the single-cell epigenetic mode will be enabled in where a gene-TF affinity matrix will be created per each column (metacell)
  - specify the columns in the bed file that shold be used to create the features.
  - STARE has in total 16 arguments including those to enable the ABC-score mode and etc. Here's an example of calling it from bash:

```
./STARE.sh -g /projects/abcpp/work/base_data/GRCh37.p13.genome.fa
-b /projects/abcpp/work/base_data/K562_10xRandomActivity.bed
-o Test_K562_10xRandom
-p ../PWMs/2.0/human_jaspar_hoc_kellis.PSEM
-a /projects/abcpp/work/base_data/gencode.v38lift37.basic.annotation.gtf
-s 5000000
-e False
-n 4+
-f /projects/abcpp/work/base_data/K562_ChrWise_Contacts/
-k 5000
-t 0
-c 10``
```

3.1 Inputs

- 1. Seurat Gene expression object (fixed category annotation: cell type, subcell type (cluster) (default)
- 2. Static TF object (gene x TF)
- 3. (optional) Bulk epigenomic data per cell type -> dynamic TF object (gene x TF x general closest cellType, eg., hepG2 and LiHe)
- 4. (optional) Single cell ATAC data -> create 3. and 4. Metacell TF object (TF X gene x metacell)
- 5. (optional) Cell matching metadata(barcode cell, ATAC barcode, RNA barcode)

3.2 Format for optional argument 3

peak bed files (TEPIC, gene annotation) metadata (cell type=file)

3.3 Format for optional argument 4

peak bed files per cell -> summarize per metacell (TEPIC, gene annotation) metadata (cell type=file) the user should point to a folder where the bed files are stored. The bed files must have meaningful tags such as barcode, etc.

- 5. **matched single cell RNA and ATAC** cell metadata (barcode cell, ATAC barcode, RNA barcode)
- **Genome file: GeneID geneSymbol GeneLength TSS Chr Start End Strand** It is used to compute the TPM conversion and features.

CHAPTER

FOUR

HYPER-PARAMETERS

The hyper-parametrs for GAZE can be divided into several parts. First set of parameters are for running the triangulate functions:

```
triang_opts <- list(file_name= "",
            RNA_count= "",
            ATAC_count= "",
            celltype= "",
            assay= "",
            umap_flag= T,
            expected_cells= 30,
            pca_components= 30,
            umap_components= 20,
            conditions= ""
            )
```

Second is the set of options for the MetaCellar package:

```
setClass("MetaCellaR.options", slots= list(umap_flag= "logical",
                                    RNA_slot= "character",
                                    ATAC_slot= "character",
                                    celltype_slot= "character",
                                    assay_slot= "character",
                                    expected_cells= "numeric",
                                    pca_components= "numeric",
                                    umap_components= "numeric",
                                    output_file= "character",
                                    condition= "character")
)
```

Next are the parameters for generating the TF affinities through *STARE* (see the full description of their parameters here).

The parameters for the GAZE object are explained in the chapter below. * Learning = {both,level2}

- Mode = {level1, all}
- filter = if T use our filtering steps implemented in the triangulate pipeline, otherwise use all the genes provided in the Seurat object

CHAPTER

FIVE

FUNCTIONS

Here you can find a list of functions implemented in GAZE that will hopefully produce what you need!

5.1 second_level_learnings

To generate the per-gene models, the function *second_level_learnings* from the *triangulate* package should be called:

```
sec_res <- second_level_learnings(TGL.model_m= tri_res$TGL.model,
partition_m= tri_res$partition, ID_mapping)
```

Write the function description

CHAPTER

SIX

GAZE OBJECT

Withing GAZE we create an R object that holds many different information, such as number of metacells (if computing metacells was enabled), number of genes, glmnet objects of the regression models, etc. This allows the user to access the elements in this object for their customized applications. But first, we need to explain `GAZE.options` class and its attributes. GAZE options class structure:

```
setClass("GAZE.options", slots= list(TPM= "logical",
                              MetacellaR= "logical",
                              genome_version= "character", ## MUST
                              organism= "character", ## MUST
                              gtf_file= "character", ## MUST
                              ATAC_exists= "logical",
                              filtering= "logical",
                              rna_file= "character",
                              atac_file= "character")
)
```

The *genome_version*, *organism*, and *gtf_file* are mandatory. The *rna_file`(also mandatory) and `atac_file* (optional) are the paths to the RNA and ATAC data. respectively.

GAZE class structure:

```
setClass("GAZE", slots= list(STARE_options= "STARE.options",
                      triang_options= "list",
                      gaze.options= "GAZE.options",
                      seurat_file= "character",
                      expr_mat= "dgCMatrix",
                      metacell_mat= "data.frame",
                      RNA_MC_map= "data.frame",
                      condition= "character",
                      celltype= "character",
                      ATAC_mat= "matrix",
                      ATAC_peaks= "character",
                      lost_genes= "data.frame",
                      lost_TFs= "data.frame",
                      models= "list",
                      shap_values= "list",
                      ID_mapping= "data.frame",
                      TF_cell_mat= "matrix")
)
```

After having the *STARE*, *triangulate*, and *GAZE* options set, you are good to go constructing the GAZE object. As shown above, it contains those three arguments for handling the general options for the individual modules of the tool, as

well as arguments such as *seurat_file* that points to the path where the Seurat object is stored. Throughout the workflow, other slots will get their values. For example, if the *MetaCellaR* flag was enabled, the corresponding function will be run on the sc-RNA data and the aggregated cells (metacells) will be stored in the metacell_mat and the mapping of the individual cell barcodes to the metacells will be stored in RNA_MC_map. *expr_mat* is a sparse matrix that holds the gene expression for the individual cells. This is used to compare how the gene expression profile differs among the individual and aggregated (metacell) cells.

There are other slots in the GAZE object that refer to the cell *condition* or *celltype*, which they normally will be inferred from the Suerat object. In case the ATAC data was accompanied by the RNA data, the following two slots will be updated through the construction of the GAZE object: 1) *ATAC_mat* and *ATAC_peaks*. As the name suggests, *ATAC_mat* contains a matrix of ATAC reads measured in peaks that are stored in *ATAC_peaks*.

In addition, there are two other slots that are meant to keep track of missing genes or transciprtion factors (TFs): *lost_genes* and *lost_TFs*. The genes might be lost during some filtering steps that require discarding lowly expressed genes across cells or when the first level learning is enabled, the matching of training instances (genes) between gene expression data and TF binding affinities may lead to removal of some genes. TFs, on the other hand, can be lost when the corresponding gene (gene that has the same symbol as the TF name) is missing among the features and response variables.

The slot *models* hold the glmnet models derived for each individual gene and *shap_values* has the SHAP values for each of these models. We also obtained a mapping of gene Ensembl IDs to gene symbols from the *gtf* file that the user has provided (`@gaze.options@gtf_file`). This mapping can be accessed from the GAZE object using the @ operator as follows: `@ID_mapping`.

CHAPTER

SEVEN

VISUALIZATION

A set of plots will automatically be created in the output folder provided by the user, however we have developed a ShinyApp that delivers many layers of visualization starting from exploring attributes of the data all the way to pairwise comparisons of different aspects of regulation across cell types or conditions.