scMarco

Visualize binarized gene expression for combinatorial marker discovery

Yu-Chieh David Chen *et al.*

2023-04-02

Table of contents

1 Introduction **3**

1.1 Contact . . . 3

1.2 References . . . 3

2 Prepare Data **4**

3 Configuration **9**

3.1 `title` . . . 9

3.2 `database_path` . . . 10

3.3 `databases` . . . 10

3.4 `gene_list` . . . 10

4 Quick Start **11**

4.1 Recommended Workflow . . . 11

4.2 Select your first marker . . . 12

4.3 Select a second marker . . . 12

4.4 Inspect your marker combo . . . 12

5 Select the First Marker **13**

6 Shared Options **14**

6.1 Which dataset to use? . . . 14

6.1.1 From cluster . . . 14

6.1.2 From gene . . . 15

6.1.3 Find distinct genes within . . . 15

6.1.4 Co-expression plot . . . 15

6.1.5 Plot expression trend . . . 15

1 Introduction

`scMarco` (Chen et al. 2023) aims to streamline the selection of actionable marker combinations with a friendly exploration interface powered by a Bayesian mixture model (Davis et al. 2020; Özel et al. 2021) to binarize gene expression in scRNA-seq.

`scMarco` can be run locally on your personal computer if you have `R` and supporting packages including `Shiny` installed. It can also be deployed on the internet like any Shiny app if you wish to provide your datasets to the public.

To prepare your data for `scMarco`, we provide a `Snakemake` workflow to discover and binarize genes that are bimodal at per-cluster level, and instructions to convert binarized and normalized expression matrices to be imported to `scMarco`.

Once `scMarco` is configured, we recommend users to start **from a cluster (From cluster)** or **from a gene that marks several clusters (From gene)** of interest. Either tab will help you identify other clusters that could be marked when you only use one marker, and help you select another marker to specifically label it (**Find distinct genes within**).

Once marker combinations are found, you might want to see how other clusters in the same dataset express this combination (**Co-exprssion plot**).

We acknowledge that while binarization greatly reduces computational complexity, it has its limitation: Gene expression is a spectrum, and there would be clusters that are hard to categorize to either ON or OFF state. We thus encourage users to also examine log-normalized expression value and its dynamic (**Plot expression trend**) and decide whether a satisfactory pair of markers have been identified.

1.1 Contact

If you have any questions, please contact Yu-Chieh David Chen about suggestions and inquiries about fly lines and genetics; if you have questions, suggestions, or bugs noted about `scMarco`, please contact Yen-Chung Chen.

1.2 References

2 Prepare Data

If you have **average normalized expression** and **modeled expression probability** for each cluster, this section will demonstrate how to prepare them for visualization with `scMarco`.

If you use Seurat to process your data, cluster average expression can be retrieved with `AverageExpression()` [1].

This will render a data.frame where each row is a gene while each column is a cluster:

```r
# A subset of data retrieved from Ozel et al. (2021)
lognorm <- read.csv(
  "../../example/data/log_norm/Adult.csv",
  row.names = 1,
  check.names = FALSE
)

head(lognorm)
```

```
           1 8    9 12 14 15 19 27 31
CG45784 0.00 0 0.00  0  0  0  0  0  0
CG45783 0.00 0 0.00  0  0  0  0  0  0
spok    0.00 0 0.00  0  0  0  0  0  0
Parp    1.00 1 0.99  1  1  1  1  1  1
Alg-2   0.00 0 0.00  0  0  0  0  0  0
Tim17b  0.69 1 1.00  1  1  1  1  1  1
```

`scMarco` uses a slightly different organization: A data.frame with 4 columns:

- gene: Gene symbols or IDs
- cluster: Cluster names or IDs
- value: The value of average expression or expression probability
- stage: The stage or condition
- type: `lognorm` for average expression; `prob` for expression probability

[1] Note that you need to set `return.seurat = TRUE` to get log-normalized average (Also see).

We can convert the above table with tidyr:

```
library(tidyr)

# Since tidyr does not deal with row names, we need to keep genes as a column
lognorm$gene <- row.names(lognorm)

lognorm_pivot <- pivot_longer(
  lognorm,
  cols = -gene, # Do not pivot genes
  values_to = "value",
  names_to = "cluster"
)

head(lognorm_pivot)
```

```
# A tibble: 6 x 3
  gene    cluster value
  <chr>   <chr>   <dbl>
1 CG45784 1           0
2 CG45784 8           0
3 CG45784 9           0
4 CG45784 12          0
5 CG45784 14          0
6 CG45784 15          0
```

Once the conversion is completed, we annotate the stage (`Adult`) and data type (`lognorm`):

```
lognorm_pivot$stage <- "Adult"
lognorm_pivot$type <- "lognorm"

head(lognorm_pivot)
```

```
# A tibble: 6 x 5
  gene    cluster value stage type
  <chr>   <chr>   <dbl> <chr> <chr>
1 CG45784 1           0 Adult lognorm
2 CG45784 8           0 Adult lognorm
3 CG45784 9           0 Adult lognorm
4 CG45784 12          0 Adult lognorm
5 CG45784 14          0 Adult lognorm
```

```
6 CG45784 15          0 Adult lognorm
```

To preserve memory usage with a large dataset, `scMarco` stores data in a SQLite database, but no worries – this will just be two extra lines in `R`.

To interact with `SQLite`, we need two extra packages: `DBI` and `RSQLite`.

```r
library(DBI)
library(RSQLite)

# This will create a new database if the file does not exist yet while
# connect to it if it exists already.
db <- dbConnect(SQLite(), "../../example/ex_db.sqlite")
```

`scMarco` supports dealing with multiple datasets as long as you give each of them a name. Here, we are going to call it `example_optic_lobe`.

If this is a new database and the table does not exist in your database, we can create the table with `dbWriteTable()`.

```r
dbWriteTable(
  conn = db, # The database you just opened/connected
  name = "example_optic_lobe",
  value = lognorm_pivot,
  # Regular user does not need this option below.
  # It is turned on to allow rebuilding the documentation without deleting
  # previously generated examples.
  overwrite = TRUE
)
```

Now, we have the average expression data in the database. Similar process can be done with expression probability:

```r
# A subset of data retrieved from Ozel et al. (2021)
mm <- read.csv(
  "../../example/data/mixture_model/Adult.csv",
  row.names = 1,
  check.names = FALSE
)

head(mm)
```

```
           1 8    9 12 14 15 19 27 31
```

```
CG45784 0.00 0 0.00  0  0  0  0  0  0
CG45783 0.00 0 0.00  0  0  0  0  0  0
spok    0.00 0 0.00  0  0  0  0  0  0
Parp    1.00 1 0.99  1  1  1  1  1  1
Alg-2   0.00 0 0.00  0  0  0  0  0  0
Tim17b  0.69 1 1.00  1  1  1  1  1  1
```

We convert and annotate the probability table as described above:

```r
# Since tidyr does not deal with row names, we need to keep genes as a column
mm$gene <- row.names(mm)

mm_pivot <- pivot_longer(
  mm,
  cols = -gene, # Do not pivot genes
  values_to = "value",
  names_to = "cluster"
)

mm_pivot$stage <- "Adult"
mm_pivot$type <- "prob"

head(mm_pivot)
```

```
# A tibble: 6 x 5
  gene    cluster value stage type 
  <chr>   <chr>   <dbl> <chr> <chr>
1 CG45784 1           0 Adult prob 
2 CG45784 8           0 Adult prob 
3 CG45784 9           0 Adult prob 
4 CG45784 12          0 Adult prob 
5 CG45784 14          0 Adult prob 
6 CG45784 15          0 Adult prob 
```

To store expression probability to an existing table (`example_optic_lobe`) that we just created, we use `dbAppendTable()` instead to append data.

```r
dbAppendTable(
  conn = db, # The database you just opened/connected
  name = "example_optic_lobe",
  value = mm_pivot
)
```

```
[1] 108252
```

Now, we have a database ready for `scMarko`. If you have multiple stages or conditions, you need to repeat the above process and append all data into the same table.

Once we are done, we can close the connection to the database by `dbDisconnect()`.

```
dbDisconnect(db)
```

3 Configuration

Once you have prepared a database, you need to point `scMarco` to it by editing a configuration file (`config.yaml`) that you can find in the directory.

The configuration file would contain several parts which we will discuss in this section.

Each part is defined by a name and a value separated by a colon(e.g., `name: value`), and indentation is used to show hierarchy. For example, `example_optic_lobe` sits under and belongs to `databases`, while `flybase_tf` is a part of `gene_list`.

```yaml
title: "scMarco"

database_path: 'example/min_db.sqlite'

databases:
  example:
    label: "Subset of optic lobe atlas from Ozel et al. (2021)"
    stages: ['P15', 'P30', 'P40', 'P50', 'P70', 'Adult']

gene_list:
  flybase_tf:
    path: "data/flybase_tfs.txt"
    label: "FlyBase TFs"
  mimic:
    path: "data/mimic_temp.txt"
    label: "RMCE swappable gene-specific lines"
```

3.1 `title`

This defines the title that you will see when you run `scMarco`.

3.2 `database_path`

This is the file path to the database you generated when you preprocessed your dataset. The path should be relative to the home directory of the app (e.g., if you have `scMarko` on your desktop (`~/Desktop/scmarko/`) and a database (`db.sqlite`) is stored in a subdirectory named `my_dbs`, you should set `database_path` to `my_dbs/db.sqlite`).

3.3 `databases`

Under this part, each item (e.g., `example`) is a dataset. The items are the table names you set when you preprocessed your dataset.

`label` defines how this dataset is referred to in the app. `stages` defines the order of stages [1].

3.4 `gene_list`

There are tens of thousands of genes in a dataset, and oftentimes we are only interested in some of them. Only keeping the subset of genes helps with the efficiently of the app. You may curate and provide lists of gene symbols as `txt` files in which each line is a gene symbol.

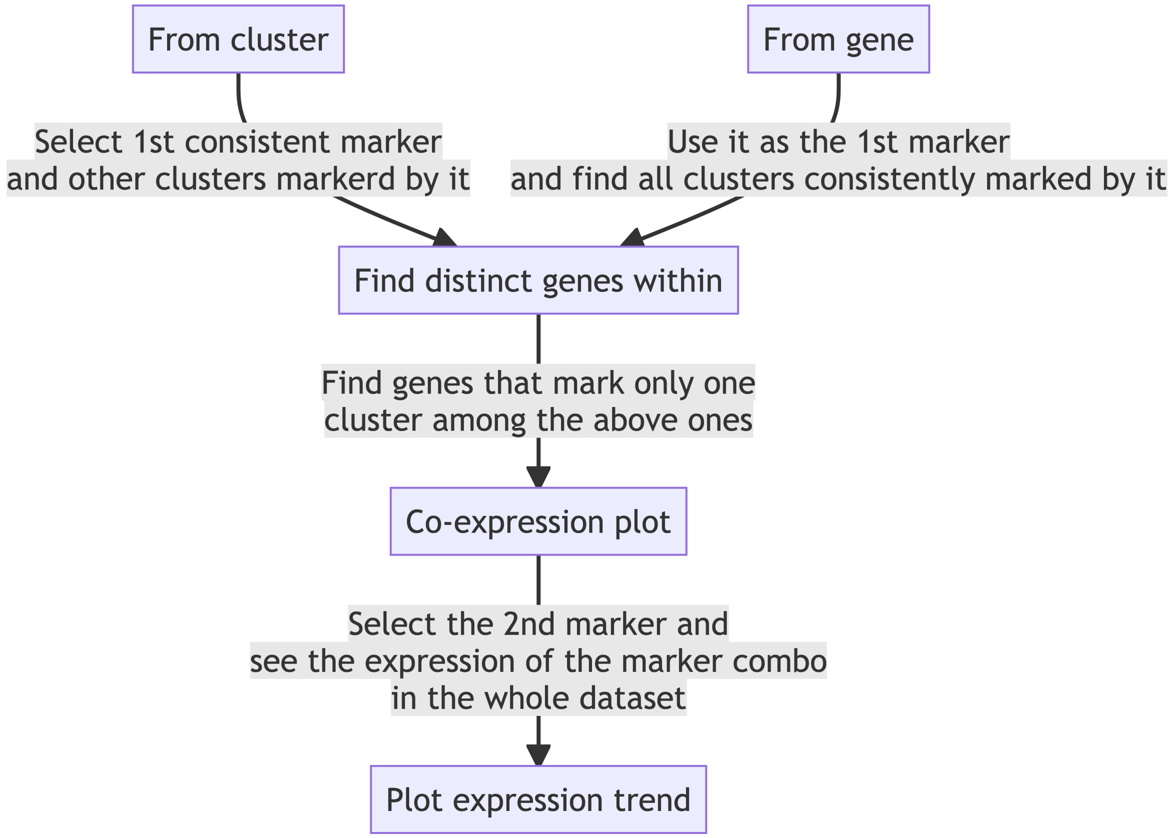
In `config.yaml`, you give a short name (e.g., `flybase_tf` and `mimic`) to your list, and provide the relative path in `path` and how the list is referred to in the app in `label`.

[1] `stages` is case sensitive and must contain the same values you have in the `stage` column in your database (Also see).

4 Quick Start

scMarko aims to help with the heuristic selection of markers for your cell types of interest by:

4.1 Recommended Workflow


From cluster
From gene
Select 1st consistent marker
and other clusters markerd by it
Use it as the 1st marker
and find all clusters consistently marked by it
Find distinct genes within
Find genes that mark only one
cluster among the above ones
Co-expression plot
Select the 2nd marker and
see the expression of the marker combo
in the whole dataset
Plot expression trend

4.2 Select your first marker

Depends on the prior knowledge about the diversity in your tissue of interest, you might start from a cell type (**From cluster**) and select a gene that marks it. `scMarko` will provide a list of other clusters (*offtargets*) that the second marker in the combination aims to distinguish.

Sometimes, you are interested in cell types that are marked by the same gene, and that gene will be your first marker. `scMarko` will list all clusters that are likely to express this gene so you can find putative second markers that will further tell them apart (**From gene**).

4.3 Select a second marker

Find distinct genes lists the genes that are expressed in only one of the clusters from **From cluster** or **From gene**. We hope the clusters that you are interested in have at least one such gene. If not, you could select the clusters that co-express your first two markers, and run **Find distinct genes** on them to find a third marker that distinguish them.

4.4 Inspect your marker combo

While the marker combinations selected by this process should be specific among the dataset, there could be gray zones during the binarization process. To see if this is the case, plotting normalized expression value and see how your clusters of interest compare to others on this scale could provide some insight.

Plot expression trend plots normalized expression values as a line plot across stages/conditions and highlights the cluster you selected.

To inspect how your marker combinations look like in your dataset (e.g., which clusters expresses either of them at a stage/condition), **Co-expression plot** plots this information for your combinations of markers.

5 Select the First Marker

6 Shared Options

6.1 Which dataset to use?

This tool retrieves information stored in a SQLite3 database, which is preprocessed by Bayesian mixture modeling from single-cell transcriptomics data.

Multiple datasets could co-exist in the database, and this switch allows users to explore what intrigues them the most.

6.1.0.1 Which genes to consider?

By default, every gene that is detected in the dataset is used for analysis and visualization. You can provide your own gene list to focus on them. Please note that gene symbols used in reference genomes could change over time, and you **need to examine if your provided gene list utilize the same naming scheme as your data**.

6.1.0.2 Clusters of interest

The clusters to focus on (Find distinct genes within) or to highlight (Plot expression trend).

6.1.0.3 Probability threshold to be considered as expressed

Binomial probability of a gene being expressed in a cluster/cell type . A cutoff is selected here to decide the probability threshold to consider a gene **ON**.

6.1.1 From cluster

Starting from one cluster, define how consistent among different stages/ conditions you desire for a marker. After clicking **Show genes**, the consistently expressing genes would be demonstrated and ranked by how many other clusters they are also detected in.

You can select a gene, and move forward to **Find distinct genes within**.

6.1.2 From gene

Similar to **From clustser**, but starting from a gene. You define how consistent among different stages/conditions you desire for a marker. After clicking **Show clusters**, all clusters that consistently express this gene would be shown, and you can choose the clusters that you want to distinguish and move forward to **Find distinct genes within**. other clusters they are also detected in.

6.1.3 Find distinct genes within

Among your clusters of interest, find genes that are only **ON** in one cluster but not others. Use the slide bar to define **ON** as *expressed in at least N stages*. Alternatively, use the check boxes to select the stages a gene must be expressed to be considered as **ON**.

Click **Find genes** to visualize how well you can distinguish these clusters with the addition of another gene. You could pick a gene from this list to use together with what you chose in **From cluster** or **From gene**.

You might want to visualize this gene pair to see how they express in other clusters in the dataset with **Co-expression plot**.

6.1.4 Co-expression plot

Select genes that you are interested in, and visualize their co-expression pattern through different stages.

6.1.5 Plot expression trend

This tool generates an interactive line plot of log-transformed cluster average normalized expression. Cluster label will be shown when you point your cursor to a point. Clusters of interest will be highlighted (and coded with distinct colors when you download a static version of the plot).

Chen, Yu-Chieh David, Yen-Chung Chen, Raghuvanshi Rajesh, Nathalie Shoji, Maisha Jacy, Haluk Lacin, Ted Erclik, and Claude Desplan. 2023. "Using Single-Cell RNA Sequencing to Generate Cell-Type-Specific Split-GAL4 Reagents Throughout Development." *bioRxiv*. https://doi.org/10.1101/2023.02.03.527019.

Davis, Fred P, Aljoscha Nern, Serge Picard, Michael B Reiser, Gerald M Rubin, Sean R Eddy, and Gilbert L Henry. 2020. "A genetic, genomic, and computational resource for exploring neural circuit function." *eLife* 9 (January): e50901. https://doi.org/10.7554/eLife.50901.

Özel, Mehmet Neset, Félix Simon, Shadi Jafari, Isabel Holguera, Yen-Chung Chen, Najate Benhra, Rana Naja El-Danaf, et al. 2021. "Neuronal Diversity and Convergence in a Visual

System Developmental Atlas." *Nature* 589 (7840): 88–95. https://doi.org/10.1038/s41586-020-2879-3.