IGVF CRISPR Jamboree 2024: MuData Proposal

Gene Katsevich

January 29, 2024

The goal of this document is to propose a `MuData` object structure for the IGVF CRISPR Jamboree 2024. It builds off of Lucas's sample Gasperini `MuData` object and the per-guide metadata format. I will propose variants of the `MuData` structure that are appropriate as inputs and outputs for the gRNA assignment and inference modules. These structures will be exemplified by a subset of the Gasperini data, distinct from Lucas's. For each module, I will present minimal examples of `MuData` objects containing required fields, as well as more fleshed out examples containing optional fields. All of the `MuData` objects are available on GitHub.

```
import mudata as md
import pandas as pd
pd.set_option('display.max_columns', None)
```

1 gRNA assignment

1.1 gRNA assignment (required fields)

1.1.1 Input

```
grna_assignment_input = md.read_h5mu("data/gasperini_grna_assignment_input_minimal.h5mu")
grna_assignment_input
```

```
## MuData object with n_obs × n_vars = 9704 × 167
##   uns:   'moi'
##   2 modalities
##     gene:    9704 x 112
##     guide:   9704 x 55
##       var:   'targeting', 'intended_target_name'
```

The minimal input `MuData` object for gRNA assignment contains two modalities: `gene` and `guide`. The `gene` modality just needs to have a `.X` matrix containing the gene RNA UMI counts. The `guide` modality needs to have a `.X` matrix containing the gRNA UMI counts, as well as a `.var` data frame containing at least the Boolean variable `targeting` and the string `intended_target_name`:

```
grna_assignment_input['guide'].var.iloc[[0, 1, 20, 21, 30, 31]]
```

```
##                      targeting intended_target_name
## ATGTAGAAGGAGACACCGGG      TRUE      ENSG00000012660
## GCGCAGAGGCGGATGTAGAG      TRUE      ENSG00000012660
## ACACCCTCATTAGAACCCAG      TRUE      candidate_enh_1
## TTAAGAGCCTCGGTTCCCCT      TRUE      candidate_enh_1
## GACCTCCTGTGATCAGGTGG     FALSE        non-targeting
## ATTGGTATCCGTATAAGCAG     FALSE        non-targeting
```

Note that the `targeting` column is a string rather than a Boolean due to type compatibility issues involving

R, Python, and HDF5. It can be cast to a Boolean if desired. Finally, the gRNA assignment input `MuData` object must contain an `uns` field called `moi` (`low` or `high`).

1.1.2 Output

```
grna_assignment_output = md.read_h5mu("data/gasperini_grna_assignment_output_minimal.h5mu")
grna_assignment_output
```

```
## MuData object with n_obs × n_vars = 9704 × 167
##   uns:   'moi'
##   2 modalities
##     gene:    9704 x 112
##     guide:   9704 x 55
##       var:   'targeting', 'intended_target_name'
##       layers:    'grna_assignments'
```

The minimal gRNA assignment output `MuData` object is the same as the input object, except it now has a `.layer` called `grna_assignments`, which is a binary assignment matrix of guides to cells.

1.2 gRNA assignment (optional fields)

1.2.1 Input

```
grna_assignment_input = md.read_h5mu("data/gasperini_grna_assignment_input.h5mu")
grna_assignment_input
```

```
## MuData object with n_obs × n_vars = 9704 × 167
##   obs:   'prep_batch', 'within_batch_chip', 'within_chip_lane'
##   uns:   'moi'
##   2 modalities
##     gene:    9704 x 112
##       obs:   'num_expressed_genes', 'total_gene_umis'
##       var:   'symbol', 'gene_chr', 'gene_start', 'gene_end'
##     guide:   9704 x 55
##       obs:   'num_expressed_grnas', 'total_grna_umis'
##       var:   'targeting', 'intended_target_name', 'intended_target_chr', 'intended_target_start', 'i
```

Optionally, the gRNA assignment input object can contain a top-level `obs` field containing cell-level information that is not specific to modality, such as batch information. Here is what it looks like for the Gasperini data:

```
grna_assignment_input.obs[['prep_batch', 'within_batch_chip', 'within_chip_lane']]
```

```
##                                     prep_batch    within_batch_chip  \
## GCTTGAATCGAATGCT-1_1B_1_SI-GA-F2  prep_batch_1  within_batch_chip_B
## AGCTTGATCGAGAGCA-1_1A_2_SI-GA-E3  prep_batch_1  within_batch_chip_A
## CCCAATCTCCTCAATT-1_1B_1_SI-GA-F2  prep_batch_1  within_batch_chip_B
## CGCGGTACACTGTCGG-1_1A_2_SI-GA-E3  prep_batch_1  within_batch_chip_A
## GGACGTCTCATGTCTT-1_1B_8_SI-GA-F9  prep_batch_1  within_batch_chip_B
## ...                                          ...                  ...
## CGCTATCTCTATCGCC-1_2A_4_SI-GA-G5  prep_batch_2  within_batch_chip_A
## TCACAAGCAGCCTTGG-1_2A_6_SI-GA-G7  prep_batch_2  within_batch_chip_A
## GCTGCAGGTGAAGGCT-1_2B_6_SI-GA-H7  prep_batch_2  within_batch_chip_B
## GGATTACCATGTTGAC-1_2A_4_SI-GA-G5  prep_batch_2  within_batch_chip_A
## GTGCTTCTCGGATGTT-1_2A_1_SI-GA-G2  prep_batch_2  within_batch_chip_A
##
##                                     within_chip_lane
```

```
## GCTTGAATCGAATGCT-1_1B_1_SI-GA-F2  within_chip_lane_1
## AGCTTGATCGAGAGCA-1_1A_2_SI-GA-E3  within_chip_lane_2
## CCCAATCTCCTCAATT-1_1B_1_SI-GA-F2  within_chip_lane_1
## CGCGGTACACTGTCGG-1_1A_2_SI-GA-E3  within_chip_lane_2
## GGACGTCTCATGTCTT-1_1B_8_SI-GA-F9  within_chip_lane_8
## ...                                          ...
## CGCTATCTCTATCGCC-1_2A_4_SI-GA-G5  within_chip_lane_4
## TCACAAGCAGCCTTGG-1_2A_6_SI-GA-G7  within_chip_lane_6
## GCTGCAGGTGAAGGCT-1_2B_6_SI-GA-H7  within_chip_lane_6
## GGATTACCATGTTGAC-1_2A_4_SI-GA-G5  within_chip_lane_4
## GTGCTTCTCGGATGTT-1_2A_1_SI-GA-G2  within_chip_lane_1
##
## [9704 rows x 3 columns]
```

The gRNA assignment input object may also include cellwise covariates for the `gene` modality, such as number of expressed genes and total RNA UMIs:

```
grna_assignment_input['gene'].obs
```

```
##                                  num_expressed_genes  total_gene_umis
## GCTTGAATCGAATGCT-1_1B_1_SI-GA-F2                   41            280.0
## AGCTTGATCGAGAGCA-1_1A_2_SI-GA-E3                   35            192.0
## CCCAATCTCCTCAATT-1_1B_1_SI-GA-F2                   41            781.0
## CGCGGTACACTGTCGG-1_1A_2_SI-GA-E3                   37            189.0
## GGACGTCTCATGTCTT-1_1B_8_SI-GA-F9                   32            262.0
## ...                                               ...              ...
## CGCTATCTCTATCGCC-1_2A_4_SI-GA-G5                   23            203.0
## TCACAAGCAGCCTTGG-1_2A_6_SI-GA-G7                   30            173.0
## GCTGCAGGTGAAGGCT-1_2B_6_SI-GA-H7                   37            428.0
## GGATTACCATGTTGAC-1_2A_4_SI-GA-G5                   47            658.0
## GTGCTTCTCGGATGTT-1_2A_1_SI-GA-G2                   23            166.0
##
## [9704 rows x 2 columns]
```

Next, the gRNA assignment input object may contain additional information about the genes:

```
grna_assignment_input['gene'].var
```

```
##                      symbol gene_chr   gene_start     gene_end
## ENSG00000008853    RHOBTB2     chr8   22844930.0   22844931.0
## ENSG00000104679     R3HCC1     chr8   23145421.0   23145422.0
## ENSG00000104689  TNFRSF10A     chr8   23082573.0   23082574.0
## ENSG00000120889  TNFRSF10B     chr8   22926533.0   22926534.0
## ENSG00000120896     SORBS3     chr8   22409208.0   22409209.0
## ...                    ...      ...          ...          ...
## ENSG00000114850       SSR3     chr3  156271913.0  156271914.0
## ENSG00000072274       TFRC     chr3  195808960.0  195808961.0
## ENSG00000134851    TMEM165     chr4   56262124.0   56262125.0
## ENSG00000198899                             NaN          NaN
## ENSG00000228253                             NaN          NaN
##
## [112 rows x 4 columns]
```

Finally, the gRNA assignment input object may contain cellwise covariates for the `guide` modality and additional information about the guides beyond the two required fields `targeting` and `intended_target_name`:

```
grna_assignment_input['guide'].obs
```

```
##                                  num_expressed_grnas  total_grna_umis
## GCTTGAATCGAATGCT-1_1B_1_SI-GA-F2                    1              9.0
## AGCTTGATCGAGAGCA-1_1A_2_SI-GA-E3                    1             18.0
## CCCAATCTCCTCAATT-1_1B_1_SI-GA-F2                    1             24.0
## CGCGGTACACTGTCGG-1_1A_2_SI-GA-E3                    1             26.0
## GGACGTCTCATGTCTT-1_1B_8_SI-GA-F9                    1             12.0
## ...                                               ...              ...
## CGCTATCTCTATCGCC-1_2A_4_SI-GA-G5                    1              5.0
## TCACAAGCAGCCTTGG-1_2A_6_SI-GA-G7                    1             39.0
## GCTGCAGGTGAAGGCT-1_2B_6_SI-GA-H7                    1             21.0
## GGATTACCATGTTGAC-1_2A_4_SI-GA-G5                    1             73.0
## GTGCTTCTCGGATGTT-1_2A_1_SI-GA-G2                    1             12.0
##
## [9704 rows x 2 columns]
```

```
grna_assignment_input['guide'].var.iloc[[0, 1, 20, 21, 30, 31]]
```

```
##                      targeting intended_target_name intended_target_chr  \
## ATGTAGAAGGAGACACCGGG      TRUE      ENSG00000012660                 chr6
## GCGCAGAGGCGGATGTAGAG      TRUE      ENSG00000012660                 chr6
## ACACCCTCATTAGAACCCAG      TRUE      candidate_enh_1                 chr8
## TTAAGAGCCTCGGTTCCCCT      TRUE      candidate_enh_1                 chr8
## GACCTCCTGTGATCAGGTGG     FALSE        non-targeting
## ATTGGTATCCGTATAAGCAG     FALSE        non-targeting
##
##                       intended_target_start  intended_target_end
## ATGTAGAAGGAGACACCGGG             53213723.0           53213738.0
## GCGCAGAGGCGGATGTAGAG             53213738.0           53213754.0
## ACACCCTCATTAGAACCCAG             23366136.0           23366564.0
## TTAAGAGCCTCGGTTCCCCT             23366564.0           23366992.0
## GACCTCCTGTGATCAGGTGG                  -9.0                 -9.0
## ATTGGTATCCGTATAAGCAG                  -9.0                 -9.0
```

2 Inference

2.1 Inference (required fields)

2.1.1 Input

Here is an example of a `MuData` object containing the minimal required fields for input to the inference module:

```
inference_input = md.read_h5mu("data/gasperini_inference_input_minimal.h5mu")
inference_input
```

```
## MuData object with n_obs × n_vars = 9704 × 167
##   uns:   'moi', 'pairs_to_test'
##   2 modalities
##     gene:    9704 x 112
##     guide:   9704 x 55
##       var:   'targeting', 'intended_target_name'
##       layers:    'grna_assignments'
```

This is the same as the minimal set of required fields for the output of the gRNA assignment module, except

there is an extra field in `uns` called `pairs_to_test`:

```
pd.DataFrame(inference_input.uns['pairs_to_test'])
```

```
##              gene_id intended_target_name
## 0    ENSG00000012660      ENSG00000012660
## 1    ENSG00000072274      ENSG00000072274
## 2    ENSG00000113552      ENSG00000113552
## 3    ENSG00000114850      ENSG00000114850
## 4    ENSG00000122644      ENSG00000122644
## ..               ...                  ...
## 127  ENSG00000160293      candidate_enh_4
## 128  ENSG00000119125      candidate_enh_5
## 129  ENSG00000107372      candidate_enh_5
## 130  ENSG00000165092      candidate_enh_5
## 131  ENSG00000135046      candidate_enh_5
##
## [132 rows x 2 columns]
```

The minimal required fields in `pairs_to_test` are `gene_id` and `intended_target_name`. Each row specifies a test to be conducted between CRISPR perturbation of a given target and the the expression of a given gene.

2.1.2 Output

```
inference_output = md.read_h5mu("data/gasperini_inference_output_minimal.h5mu")
inference_output
```

```
## MuData object with n_obs × n_vars = 9704 × 167
##   uns:   'moi', 'pairs_to_test', 'test_results'
##   2 modalities
##     gene:    9704 x 112
##     guide:   9704 x 55
##       var:   'targeting', 'intended_target_name'
##       layers:    'grna_assignments'
```

The minimal required fields in the output of the inference module are the same as the minimal required fields for the input, except there is an extra field in `uns` called `test_results`:

```
pd.DataFrame(inference_output.uns['test_results'])
```

```
##              gene_id intended_target_name       p_value
## 0    ENSG00000012660      ENSG00000012660  1.594670e-31
## 1    ENSG00000072274      ENSG00000072274  7.579728e-04
## 2    ENSG00000113552      ENSG00000113552  1.925930e-32
## 3    ENSG00000114850      ENSG00000114850  5.396521e-61
## 4    ENSG00000122644      ENSG00000122644  3.148987e-10
## ..               ...                  ...           ...
## 127  ENSG00000160293      candidate_enh_4  1.700000e-01
## 128  ENSG00000119125      candidate_enh_5  8.260000e-01
## 129  ENSG00000107372      candidate_enh_5  4.120000e-01
## 130  ENSG00000165092      candidate_enh_5           NaN
## 131  ENSG00000135046      candidate_enh_5  2.910228e-06
##
## [132 rows x 3 columns]
```

This is a data frame containing the same columns as the `pairs_to_test` data frame, plus at least one column containing a measure of the association for each pair. These columns can be `p_value`, `log2_FC`,

`posterior_probability`, or any other measure of association. We just have to standardize these column names.

2.2 Inference (optional fields)

2.2.1 Input

Here is an example of a `MuData` object containing some optional fields for input to the inference module:

```
inference_input = md.read_h5mu("data/gasperini_inference_input.h5mu")
inference_input
```

```
## MuData object with n_obs × n_vars = 9704 × 167
##   obs:   'prep_batch', 'within_batch_chip', 'within_chip_lane'
##   uns:   'moi', 'pairs_to_test'
##   2 modalities
##     gene:    9704 x 112
##       obs:   'num_expressed_genes', 'total_gene_umis'
##       var:   'symbol', 'gene_chr', 'gene_start', 'gene_end'
##     guide:   9704 x 55
##       obs:   'num_expressed_grnas', 'total_grna_umis'
##       var:   'targeting', 'intended_target_name', 'intended_target_chr', 'intended_target_start', 'i
##       layers:    'grna_assignments'
```

The additional fields are the same as those described for the gRNA assignment module. The only additional optional field is in the `pairs_to_test` data frame:

```
pd.DataFrame(inference_input.uns['pairs_to_test'])
```

```
##              gene_id intended_target_name         pair_type
## 0    ENSG00000012660      ENSG00000012660  positive_control
## 1    ENSG00000072274      ENSG00000072274  positive_control
## 2    ENSG00000113552      ENSG00000113552  positive_control
## 3    ENSG00000114850      ENSG00000114850  positive_control
## 4    ENSG00000122644      ENSG00000122644  positive_control
## ..               ...                  ...               ...
## 127  ENSG00000160293      candidate_enh_4         discovery
## 128  ENSG00000119125      candidate_enh_5         discovery
## 129  ENSG00000107372      candidate_enh_5         discovery
## 130  ENSG00000165092      candidate_enh_5         discovery
## 131  ENSG00000135046      candidate_enh_5         discovery
##
## [132 rows x 3 columns]
```

Note the third column: `pair_type`. This optional column classifies pairs based on whether they are intended to be positive controls (an association is known to exist), negative controls (an association is known not to exist), or discovery pairs (pairs where it is unknown whether an association exists). This information need not be used by the inference module, but it is useful for downstream analysis.

2.2.2 Output

Here is an example of a `MuData` object containing some optional fields for output from the inference module:

```
inference_output = md.read_h5mu("data/gasperini_inference_output.h5mu")
inference_output
```

```
## MuData object with n_obs × n_vars = 9704 × 167
##   obs:   'prep_batch', 'within_batch_chip', 'within_chip_lane'
```

```
##   uns:   'moi', 'pairs_to_test', 'test_results'
##   2 modalities
##     gene:    9704 x 112
##       obs:   'num_expressed_genes', 'total_gene_umis'
##       var:   'symbol', 'gene_chr', 'gene_start', 'gene_end'
##     guide:   9704 x 55
##       obs:   'num_expressed_grnas', 'total_grna_umis'
##       var:   'targeting', 'intended_target_name', 'intended_target_chr', 'intended_target_start', 'i
##       layers:    'grna_assignments'
```

The only difference from before is in `test_results`:

```
pd.DataFrame(inference_output.uns['test_results'])
```

```
##              gene_id intended_target_name   log2_fc       p_value  \
## 0    ENSG00000012660      ENSG00000012660 -1.055540  1.594670e-31
## 1    ENSG00000072274      ENSG00000072274 -0.220807  7.579728e-04
## 2    ENSG00000113552      ENSG00000113552 -1.271358  1.925930e-32
## 3    ENSG00000114850      ENSG00000114850 -1.363039  5.396521e-61
## 4    ENSG00000122644      ENSG00000122644 -0.308234  3.148987e-10
## ..               ...                  ...       ...           ...
## 127  ENSG00000160293      candidate_enh_4 -0.350800  1.700000e-01
## 128  ENSG00000119125      candidate_enh_5  0.357399  8.260000e-01
## 129  ENSG00000107372      candidate_enh_5 -0.025818  4.120000e-01
## 130  ENSG00000165092      candidate_enh_5       NaN           NaN
## 131  ENSG00000135046      candidate_enh_5 -0.700229  2.910228e-06
##
##             pair_type
## 0    positive_control
## 1    positive_control
## 2    positive_control
## 3    positive_control
## 4    positive_control
## ..                ...
## 127         discovery
## 128         discovery
## 129         discovery
## 130         discovery
## 131         discovery
##
## [132 rows x 5 columns]
```

Now, the output includes the optional `pair_type` as well as a `log2_fc` in addition to the `p_value` column. This illustrates how an inference method may output multiple measures of association for each pair.