Filter W4M data by sample class or feature attributes

Arthur Eschenlauer

16 December 2019

Contents

Purpose of the `w4mclassfilter` package

The purpose of the w4mclassfilter R package is to provide the computational back-end of the Galaxy tool `W4M Data Subset` (https://github.com/HegemanLab/w4mclassfilter_galaxy_wrapper).

This package (and the Galaxy tool) perform several steps, either to reduce the number samples or features to be analyzed, or to address several data issues that may impede downstream statistical analysis:

- Features that are missing from either `variableMetadata` or `dataMatrix` are eliminated.
- Features may be eliminated by specifying minimum or maximum value (or both) allowable in columns of `variableMetadata`.
- Features may be eliminated by specifying minimum or maximum intensity (or both) allowable in columns of `dataMatrix` for at least one sample for each feature ("range of row-maximum for each feature").
- Samples that are missing from either `sampleMetadata` or `dataMatrix` are eliminated.
- Samples may also be eliminated by criteria specified in a "sample class" column in `sampleMetadata`.
- Functions for imputing missing values in `dataMatrix` include:
  - imputing to zero (the default),
  - imputing to the median intensity observed for the feature,
  - making no imputation, or
  - applying a user-supplied alternative imputation function.
- Features and samples that have zero variance are eliminated.
- Samples and features have consistent order in `variableMetadata`, `sampleMetadata`, and `dataMatrix`.
  - The column for sorting `variableMetadata` or `sampleMetadata` may be specified or defaults to the first column.
- By default, the name of the first column `sampleMetadata` is set to `"sampleMetadata"`.
- The name of the first column `variableMetadata` is set to `"variableMetadata"`.
- An advanced option enables producing a "center" for each treatment in lieu of the samples for each treatment.

How the `w4m_filter_by_sample_class` function is used

Ordinarily, a Galaxy tool wrapper invokes `w4m_filter_by_sample_class`. For exploratory or debugging purposes, the package may be installed loaded into R and help may then be obtained with the following command:

`?w4mclassfilter::w4m_filter_by_sample_class`

W4M uses the XCMS and CAMERA packages to preprocess GC-MS or LC-MS data, producing three files that are documented in detail on the Workflow4Metabolomics (W4M) web site. In summary:

- `sampleMetadata.tsv`: a tab-separated file with metadata for the samples, one line per sample:
  - One column of this file indicates the *class* of the sample.
    * This is the class that is used by this function to determine whether to include the sample in, or to exclude the sample from, further analysis.
    * The user may use the `class_column` parameter to specify the class column.
- `variableMetadata.tsv`: a tab-separated file with metadata for the features detected, one line per feature:
  - A feature is a location in the two dimensional space defined by the GC-MS or LC-MS data set, which corresponds to a compound or a group of compounds.
  - One dimension is the mass-to-charge ratio, `m/z`.
  - The other dimension is the `retention time`, i.e., how long until the solvent gradient eluted the compound(s) from the column.
- `dataMatrix.tsv`: a tab separated file with the MS intensities for each sample for each feature:
  - There is one column per sample.
  - There is one row per feature.
  - If a feature is missing for a sample, the intensity value is `NA`.
  - For numerical reasons, intenisities may be negative, but this has no meaning in the real world.

Input- and Output-Format

Ordinary usage of the `w4mclassfilter::w4m_filter_by_sample_class` method is to read from and write to tab-delimited flat files (TSVs) because Galaxy presents datasets to tools as files. However, because general-purpose R packages usually use data structures in memory for their input and output, this function can accept not only with TSVs but also with data structures (data.frame, matrix, list, env); see 'Flexible Input and Output' below for details.

For all inputs and outputs that are file paths, those paths must be unique.

Feature- and Sample-Elimination

When `w4m_filter_by_sample_class` is invoked:

- an array of class names may be supplied in the `classes` argument. If the `include` argument is true, then only samples whose *class* column in `sampleMetadata` (as named in the `class_column` argument) will be *included in* the output; by contrast, if the `include` argument is false, then only samples whose *class* column in `sampleMetadata` will be *excluded from* the output.

  - Class names may be explicit, or regular expressions may be used.
  - See 'Support for Regular Expressions' below for details on using regular expressions.

- an array of range specification strings may be supplied in the `variable_range_filter` argument. If supplied, only features having numerical values in the specified column of `variableMetadata` that fall within the specified ranges will be retained in the output. Each range is a string of three colon-separated values (e.g., "mz:200:800") in the following order:

  - the name of a column of `variableMetadata` which must have numerical data (e.g., "mz");
  - the minimum allowed value in that column for the feature to be retained (e.g., 200);

- the maximum allowed value (e.g., 800).
- if the "maximum" is less than the "minimum", then the range is exclusive (e.g., "mz:800:200" means retain only features whose mz is NOT in the range 200-800)
- Note for the range specification strings: if the name supplied in the first field is 'FEATMAX', then the string is defining the minimum (and possibly, though less useful, maximum) intensity for each feature in the dataMatrix. For example, "FEATMAX:1e6:" would specify that any feature would be excluded if no sample had an intensity for that feature greater than 1000000.
  - note that if the "maximum" is greater than the "minimum" then the FEATMAX range specification is ignored.

Note that even when no rows or columns of the input `dataMatrix` input have zero variance, there is the possibility that eliminating samples or features may result in some rows or columns having zero variance, adversely impacting downstream statistical analysis. Consequently, `w4m_filter_by_sample_class` eliminates these rows or columns and the corresponding rows from `sampleMetadata`, `variableMetadata`, and `dataMatrix`.

Support for Imputation of Missing Values

w4m_filter_zero_imputation

The `w4mclassfilter::w4m_filter_zero_imputation` function is the default imputation method used by `w4m_filter_by_sample_class`. This function imputes negative and `NA` intensity values as zero.

```
w4m_filter_zero_imputation <-
  function(m) {
    # replace NA values with zero
    m[is.na(m)] <- 0
    # replace negative values with zero, if applicable
    m[m<0] <- 0
    # return matrix as the result
    return (m)
  }
```

w4m_filter_median_imputation

The `w4mclassfilter::w4m_filter_median_imputation` function imputes negative intensity values as zero and `NA` intensity values as the median value for the corresponding feature.

```
w4m_filter_median_imputation <-
  function(m) {
    # Substitute NA with median for the row.
    # For W4M datamatrix:
    #   - each row has intensities for one feature
    #   - each column has intensities for one sample
    interpolate_row_median <- function(m) {
      # ref: https://stats.stackexchange.com/a/28578
      #   - Create a data.frame whose columns are features and rows are samples.
      #   - For each feature, substitute NA with the median value for the feature.
      t_result <- sapply(
          as.data.frame(t(m))
        , function(x) {
            x[is.na(x)] <- median(x, na.rm = TRUE)
            x
```

```
      }
    , simplify = TRUE
    )
  #   - Recover the rownames discarded by sapply.
  rownames(t_result) <- colnames(m)
  #   - Transform result so that rows are features and columns are samples.
  m <- t(t_result)
  # eliminate negative values
  m[m < 0] <- 0
  return (m)
}
  return (interpolate_row_median(m))
}
```

w4m_filter_no_imputation

The `w4mclassfilter::w4m_filter_no_imputation` function imputes negative intensity values as zero and leaves `NA` intensity values unaffected.

```
w4m_filter_no_imputation <-
  function(m) {
    # replace negative values with zero, if applicable
    m[m < 0] <- 0
    return (m)
  }
```

Support for Regular Expressions

`w4mclassfilter::w4m_filter_by_sample_class` supports use of R regular expression patterns to select class-names.

The R `base::grepl` function (at the core of this functionality) uses POSIX 1003.2 standard regular expressions, which allow precise pattern-matching and are exhaustively defined at:

http://pubs.opengroup.org/onlinepubs/9699919799/basedefs/V1_chap09.html

However, only a few basic building blocks of regular expressions need to be mastered for most cases:

Within square brackets:

- "^" as the first character after the left bracket specifies that none listed characters should be matched
- "-" separates a range of characters, e.g., "4-7" or "b-f".

Outside of square brackets:

- "^" matches the beginning of a class-name
- "$" matches the end of a class-name
- "." matches a single character
- "*" matches the character specified immediately before zero or more times

Caveat: The tool wrapper uses the comma (",") to split a list of sample-class names, so *commas may not be used within regular expressions for this tool*

First Example: Consider a field of class-names consisting of

marq3,marq6,marq9,marq12,front3,front6,front9,front12

4

this regular expression	matches this set of sample-class names

this regular expression	matches this set of sample-class names
`^front[0-9][0-9]*$`	"front3,front6,front9,front12"
`^[a-z][a-z]3$`	"front3,marq3"
`^[a-z][a-z]12$`	"front12,marq12"
`^[a-z][a-z][0-9]$`	"front3,front6,front9,marq3,marq6,marq9"

Second Example: Consider these regular expression patterns as possible matches to a sample-class name

      `AB0123`

this regular expression	matches this set of sample-class names
`^[A-Z][A-Z][0-9][0-9]*$`	`AB0123`
`^[A-Z][A-Z]*[0-9][0-9]*$`	`AB0123`
`^[A-Z][0-9]*`	`AB0123`, see Note 1.
`^[A-Z][A-Z][0-9]`	`AB0123`, see Note 2.
`^[A-Z][A-Z]*[0-9][0-9]$`	NO MATCH, see Note 3.
`^[A-Z][0-9]*$`	NO MATCH, see Note 4.

- Note 1. The first character is a letter, "`*`" can specify zero characters, and end of line did not need to be matched.
- Note 2. The first two characters are letters aind the third is a digit.
- Note 3. The name does not end with the pattern "`[A-Z][0-9][0-9]$`", i.e., it ends with four digits, not two.
- Note 4. The pattern specifies that second character and all those that follow, if present, must be digits.

Flexible Input and Output

To support XCMS outside the context of Galaxy, `w4mclassfilter::w4m_filter_by_sample_class` supports input from and output to data structures as follows:

Inputs:

- `dataMatrix_in`
    - *if a string*, treat as a file path (i.e., same as in previous releases)
    - *else if an env or list*, read `dataMatrix_in$dataMatrix`
    - *else if a data.frame*, use `as.matrix(dataMatrix_in)`
    - *else if a matrix*, use directly
    - *else* error
- `sampleMetadata_in`
    - *if a string*, treat as a file path (i.e., same as in previous releases)
    - *else if an env or list*, read `sampleMetadata_in$sampleMetadata`
    - *else if a data.frame*, use directly
    - *else* error
- `variableMetadata_in`
    - *if a string*, treat as a file path (i.e., same as in previous releases)
    - *else if an env or list*, read `sampleMetadata_in$sampleMetadata`
    - *else if a data.frame*, use directly
    - *else* error

Outputs:

- `dataMatrix_out`
    - *if a string*, treat as a file path (i.e., same as in previous releases)
    - *else if an env or list*, write `dataMatrix_out$dataMatrix`
    - *else* error
- `sampleMetadata_out`
    - *if a string*, treat as a file path (i.e., same as in previous releases)
    - *else if an env or list*, write `sampleMetadata_out$sampleMetadata`
    - *else* error
- `variableMetadata_out`
    - *if a string*, treat as a file path (i.e., same as in previous releases)
    - *else if an env or list*, write `sampleMetadata_out$sampleMetadata`
    - *else* error

Computing Treatment Centers

`w4mclassfilter::w4m_filter_by_sample_class` supports provides as an advanced option to compute one of three types of centers for each treatment:

- "centroid" - Return only treatment-centers computed for each treatment as the mean intensity for each feature.
- "median" - Return only treatment-centers computed for each treatment as the median intensity for each feature.
- "medoid" - Return only treatment-centers computed for each treatement as the sample most similar to the other samples, using distances computed in principal-components space.
- "none" - Return all samples; do not computing centers.