# Vignette for the *sampleClassifier* R Package

*Khadija El Amrani* and *Andreas Kurtz*

Charité - Universitätsmedizin Berlin, Berlin Brandenburg Center for Regenerative Therapies (BCRT)

*a.khadija@gmx.de*

May 1, 2024

## Contents

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>Contents of the package</td>
<td>2</td>
</tr>
<tr>
<td>2.1</td>
<td><code>classifyProfile()</code></td>
<td>2</td>
</tr>
<tr>
<td>2.1.1</td>
<td>Parameter Settings</td>
<td>2</td>
</tr>
<tr>
<td>2.1.2</td>
<td>Output</td>
<td>3</td>
</tr>
<tr>
<td>2.2</td>
<td><code>classifyProfile.rnaseq()</code></td>
<td>3</td>
</tr>
<tr>
<td>2.2.1</td>
<td>Parameter Settings</td>
<td>3</td>
</tr>
<tr>
<td>2.2.2</td>
<td>Output</td>
<td>3</td>
</tr>
<tr>
<td>2.3</td>
<td><code>classifyProfile.svm()</code></td>
<td>3</td>
</tr>
<tr>
<td>2.3.1</td>
<td>Parameter Settings</td>
<td>3</td>
</tr>
<tr>
<td>2.3.2</td>
<td>Output</td>
<td>4</td>
</tr>
<tr>
<td>2.4</td>
<td><code>classifyProfile.rnaseq.svm()</code></td>
<td>4</td>
</tr>
<tr>
<td>2.4.1</td>
<td>Parameter Settings</td>
<td>4</td>
</tr>
<tr>
<td>2.4.2</td>
<td>Output</td>
<td>4</td>
</tr>
<tr>
<td>2.5</td>
<td><code>get.heatmap()</code></td>
<td>4</td>
</tr>
<tr>
<td>2.5.1</td>
<td>Parameter Settings</td>
<td>5</td>
</tr>
<tr>
<td>2.5.2</td>
<td>Output</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>Getting Started</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>Classification using sampleClassifier</td>
<td>5</td>
</tr>
<tr>
<td>4.1</td>
<td>Classification of microarray data</td>
<td>5</td>
</tr>
<tr>
<td>4.2</td>
<td>Classification of RNA-seq data</td>
<td>9</td>
</tr>
<tr>
<td>5</td>
<td>sampleClassifier algorithm details</td>
<td>13</td>
</tr>
<tr>
<td>6</td>
<td>R sessionInfo</td>
<td>14</td>
</tr>
</tbody>
</table>
1 Introduction

Discrimination between different classes of samples such as different cell types or tissues using gene expression profiles is an important problem in genetic and cell research. It has several implications and can contribute to our understanding of cell phenotype differences and will allow precise identification of various cell types and tissues. *sampleClassifier* provides functions for the classification of samples using their gene expression profiles. The package supports the classification of microarray and RNA-seq gene expression profiles. The tool requires a reference and a test data set, and uses a simple algorithm called Shared Marker Genes (SMG). As the name suggests, the number of shared marker genes between a reference and a query sample is used as a similarity measure. Marker genes are detected using the tools *MGFM* [1, 2] or *MGFR* [3] that we have developed previously. *sampleClassifier* can be applied: i) to evaluate the similarity of experimentally derived cells with their desired target cell type (e.g. to compare primary hepatocytes with induced hepatocytes); ii) to compare in vitro derived organoids to their in vivo counterparts; iii) to classify different types of diseases.

This vignette provides an introduction to gene expression profile based sample classification using the *sampleClassifier* R package and the accompanying data package, *sampleClassifier-Data*. It contains guidelines on how to select all inputs to the tool (such as reference and test matrices), and instructions on running *sampleClassifier* using microarray and RNA-seq data from the *sampleClassifierData*.

2 Contents of the package

The *sampleClassifier* package contains the following functions:

```r
> library("sampleClassifier")
> ls("package:sampleClassifier")
[1] "classifyProfile"        "classifyProfile.rnaseq"
[3] "classifyProfile.rnaseq.svm" "classifyProfile.svm"
[5] "get.heatmap"
```

2.1 `classifyProfile()`

`classifyProfile()` is a function to classify microarray gene expression profiles.

2.1.1 Parameter Settings

1. `ref_matrix`: Normalized microarray data matrix to be used as reference, with probe sets corresponding to rows and samples corresponding to columns.

2. `query_mat`: Normalized microarray query matrix with query samples to be classified, with probe sets corresponding to rows and samples corresponding to columns.

3. `chip1`: Chip name of the reference matrix (e.g. ’hgu133plus2’).

4. `chip2`: Chip name of the query matrix. This parameter can be ignored if the reference and query matrix are from the same chip.

5. `fun1`: `mean` or `median`. This will specify the number of marker genes that will be used for classification. Default is `median`.
6. **fun2**: mean or median. This will be used to summarize the expression values of probe sets that belong to the same gene. This parameter can be ignored if the reference and query matrix are from the same chip. Default is mean.

7. **write2File**: If TRUE, the classification results for each query profile will be written to a file.

8. **out.dir**: Path to a directory to write the classification results, default is the current working directory.

### 2.1.2 Output

The function `classifyProfile()` returns a list with hits for each of the query samples in the query matrix. The hits are sorted according to their similarity to the query.

### 2.2 `classifyProfile.rnaseq()`

`classifyProfile.rnaseq()` is a function to classify RNA-seq gene expression profiles.

#### 2.2.1 Parameter Settings

1. **ref_matrix**: RNA-seq data matrix to be used as reference, with genes corresponding to rows and samples corresponding to columns.

2. **query_mat**: RNA-seq query matrix with query samples to be classified, with genes corresponding to rows and samples corresponding to columns.

3. **gene.ids.type**: Type of the used gene identifiers, the following gene identifiers are supported: ensembl, refseq and ucsc gene ids. Default is ensembl.

4. **fun1**: mean or median. This will specify the number of marker genes that will be used for classification. Default is median.

5. **write2File**: If TRUE, the classification results for each query profile will be written to a file.

6. **out.dir**: Path to a directory to write the classification results, default is the current working directory.

#### 2.2.2 Output

The function `classifyProfile.rnaseq()` returns a list with top hits for each query profile, sorted according to a similarity score.

### 2.3 `classifyProfile.svm()`

`classifyProfile.svm()` is a function to classify microarray gene expression profiles using Support Vector Machines (SVM).

#### 2.3.1 Parameter Settings

1. **ref_matrix**: Normalized microarray data matrix to be used as reference, with probe sets corresponding to rows and samples corresponding to columns.
2. **query_mat**: Normalized microarray query matrix to be classified, with probe sets corresponding to rows and samples corresponding to columns.

3. **chip1**: Chip name of the reference matrix (e.g. 'hgu133plus2').

4. **chip2**: Chip name of the query matrix. This parameter can be ignored if the reference and query matrix are from the same chip.

5. **fun1**: `mean` or `median`. This will specify the number of marker genes that will be used for classification. Default is `median`.

6. **fun2**: `mean` or `median`. This will be used to summarize the expression values of probe sets that belong to the same gene. This parameter can be ignored if the reference and query matrix are from the same chip.

### 2.3.2 Output

The function `classifyProfile.svm()` returns a data frame with the predicted classes for each query profile.

### 2.4 `classifyProfile.rnaseq.svm()`

`classifyProfile.rnaseq.svm()` is a function to classify RNA-seq gene expression profiles using Support Vector Machines (SVM).

#### 2.4.1 Parameter Settings

1. **ref_matrix**: RNA-seq data matrix to be used as reference, with genes corresponding to rows and samples corresponding to columns.

2. **query_mat**: RNA-seq query matrix with query samples to be classified, with genes corresponding to rows and samples corresponding to columns.

3. **gene.ids.type**: Type of the used gene identifiers, the following gene identifiers are supported: ensembl, refseq and ucsc gene ids. Default is ensembl.

4. **fun1**: `mean` or `median`. This will specify the number of marker genes that will be used for classification. Default is `median`.

#### 2.4.2 Output

The function `classifyProfile.rnaseq.svm()` returns a data frame with the predicted classes for each query profile.

Please note that all replicates of a sample type should have the same label in the reference matrix.

### 2.5 `get.heatmap()`

`get.heatmap()` is a function to display the classification predictions as a heatmap.
2.5.1 Parameter Settings

1. res.list: the result list returned by the function `classifyProfile()` or `classifyProfile.rnaseq()`.

2.5.2 Output

The function `get.heatmap()` is used only for the side effect of creating a heatmap.

3 Getting Started

The `sampleClassifier` package can be downloaded and installed by running the following code from within R:

```r
> if (!requireNamespace("BiocManager", quietly=TRUE))
+ install.packages("BiocManager")
> BiocManager::install("sampleClassifier")
```

We also recommend installing the accompanying data package, `sampleClassifierData`, which contains pre-processed microarray and RNA-seq data, which are derived from normal human tissues, and are available from public repositories. More details about the data can be found in the Vignette of the data package.

After installing and loading `sampleClassifierData`, the individual `sampleClassifierData` datasets can be loaded using the function `data()`. For instance, the RNA-seq dataset named `se_rnaseq_refmat` is stored as a `SummarizedExperiment` object. The numeric matrix can be extracted using the function `assay()` from the `SummarizedExperiment` R package:

```r
> library(sampleClassifierData)
> data("se_rnaseq_refmat")
> rnaseq_refmat <- assay(se_rnaseq_refmat)
```

4 Classification using `sampleClassifier`

We will use the data provided in the data package `sampleClassifierData` to demonstrate how to classify samples using `sampleClassifier`.

4.1 Classification of microarray data

To classify microarray gene expression profiles, we use the function `classifyProfile()`. It expects a reference and a test matrix. We recommend using 3 replicates for each sample type in the reference matrix. Please note that replicates of the same sample type should have the same name in the reference matrix. We also recommend processing the reference and the test matrix in the same way.

As a reference matrix we use here a microarray dataset derived from the study GSE3526 [4], which is available from GEO [5]. This dataset is named `se_micro_refmat` and can be loaded with the following code:
Vignette for the `sampleClassifier` R Package

```r
> library(sampleClassifierData)
> data("se_micro_refmat")
> micro_refmat <- assay(se_micro_refmat)
> dim(micro_refmat)
[1] 54675  78

As a test matrix we use a microarray dataset derived from the study GSE2361 [6], which is available from GEO [5]. This dataset is named `se_micro_testmat` and can be loaded with the following code:

```r
> data("se_micro_testmat")
> micro_testmat <- assay(se_micro_testmat)
> dim(micro_testmat)
[1] 22283  16
```

Now, we can call the function `classifyProfile()`:

```r
> res1.list <- classifyProfile(ref_matrix=micro_refmat, query_mat=micro_testmat,
+ chip1="hgu133plus2",chip2="hgu133a", write2File=FALSE)

The reference matrix and the query are from different platforms...
Collapse rows ...
detecting marker genes...
16 profiles to be classified...
done!

For simplicity, we show only the two top hits for each query sample:

```r
> lapply(res1.list,"[",c(1,2),,drop=FALSE)

$'GSM44671 : Heart'
   Hits Score   Ratio
 1  heart.atrium 0.544 37 / 68
 2 trigeminal.ganglia 0.231 3 / 13

$'GSM44673 : Spleen'
   Hits Score   Ratio
 1  bone.marrow 0.291 23 / 79
 2 lymph.nodes 0.271 13 / 48

$'GSM44674 : Ovary'
   Hits Score   Ratio
 1    ovary 0.368 21 / 57
 2 saphenous.vein 0.135 10 / 74

$'GSM44675 : Kidney'
   Hits Score   Ratio
 1  kidney.cortex 0.456 36 / 79
 2 trigeminal.ganglia 0.231 3 / 13

$'GSM44676 : Skeletal Muscle'
   Hits Score   Ratio
```
```
Vignette for the `sampleClassifier` R Package

1 skeletal_muscle 0.646 51 / 79
2 ovary 0.14 8 / 57

`GSM44678 : Prostate`
Hits Score Ratio
1 prostate_gland 0.241 19 / 79
2 trigeminal_ganglia 0.154 2 / 13

`GSM44689 : Cerebellum`
Hits Score Ratio
1 cerebellum 0.342 27 / 79
2 thalamus 0.3 3 / 10

`GSM44693 : Bone Marrow`
Hits Score Ratio
1 bone_marrow 0.684 54 / 79
2 lymph_nodes 0.125 6 / 48

`GSM44698 : Thalamus`
Hits Score Ratio
1 thalamus 0.7 7 / 10
2 trigeminal_ganglia 0.231 3 / 13

`GSM44699 : Pituitary Gland`
Hits Score Ratio
1 pituitary_gland 0.544 43 / 79
2 thalamus 0.2 2 / 10

`GSM44700 : Spinal Cord`
Hits Score Ratio
1 spinal_cord 0.284 21 / 74
2 dorsal_root_ganglia 0.182 6 / 33

`GSM44701 : Testis`
Hits Score Ratio
1 testes 0.911 72 / 79
2 trigeminal_ganglia 0.231 3 / 13

`GSM44702 : Liver`
Hits Score Ratio
1 liver 0.861 68 / 79
2 thalamus 0.2 2 / 10

`GSM44704 : Lung`
Hits Score Ratio
1 lung 0.556 25 / 45
2 trigeminal_ganglia 0.154 2 / 13

`GSM44705 : Fetal Lung`
Hits Score Ratio
1 lung 0.311 14 / 45
To display the classification results as a heatmap, we call the function `get.heatmap()` with the resulted list as input.

```
> get.heatmap(res1.list)
```

In order to compare the classification predictions for microarray query profiles to those of Support Vector Machines (SVM), the function `classifyProfile.svm()` was implemented based on functions from the R-package e1071.

```
> res1.svm.df <- classifyProfile.svm(ref_matrix=micro_refmat, query_mat=micro_testmat,
+ chip1="hgu133plus2",chip2="hgu133a")
```
The reference matrix and the query are from different platforms...
Collapse rows...
detecting marker genes...
building an SVM model...
16 profiles to be classified...
done!

> res1.svm.df

<table>
<thead>
<tr>
<th>query_name</th>
<th>predicted_class</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSM44671 : Heart</td>
<td>heart_atrium</td>
</tr>
<tr>
<td>GSM44673 : Spleen</td>
<td>spleen</td>
</tr>
<tr>
<td>GSM44674 : Ovary</td>
<td>ovary</td>
</tr>
<tr>
<td>GSM44675 : Kidney</td>
<td>kidney_cortex</td>
</tr>
<tr>
<td>GSM44676 : Skeletal Muscle</td>
<td>skeletal_muscle</td>
</tr>
<tr>
<td>GSM44678 : Prostate</td>
<td>urethra</td>
</tr>
<tr>
<td>GSM44689 : Cerebellum</td>
<td>cerebellum</td>
</tr>
<tr>
<td>GSM44693 : Bone Marrow</td>
<td>bone_marrow</td>
</tr>
<tr>
<td>GSM44698 : Thalamus</td>
<td>thalamus</td>
</tr>
<tr>
<td>GSM44699 : Pituitary Gland</td>
<td>pituitary_gland</td>
</tr>
<tr>
<td>GSM44700 : Spinal Cord</td>
<td>spinal_cord</td>
</tr>
<tr>
<td>GSM44701 : Testis</td>
<td>testes</td>
</tr>
<tr>
<td>GSM44702 : Liver</td>
<td>liver</td>
</tr>
<tr>
<td>GSM44704 : Lung</td>
<td>lung</td>
</tr>
<tr>
<td>GSM44705 : Fetal Lung</td>
<td>lung</td>
</tr>
<tr>
<td>GSM44706 : Fetal Liver</td>
<td>bone_marrow</td>
</tr>
</tbody>
</table>

Our method `classifyProfile()` classified 14 samples correctly, whereas SVM classified 13 of the 16 query samples correctly. The sample 'GSM44678' was classified correctly by `classifyProfile()` as prostate and miclassified by SVM as urethra. The sample 'GSM44673' (tissue: spleen) was misclassified as bone marrow or adipose tissue omental by `classifyProfile()` or `classifyProfile.svm()`, respectively.
The sample 'GSM44706' (tissue: fetal liver) was misclassified as bone marrow by SVM. `classifyProfile()` predicted both bone marrow and liver as top hits with the same similarity score.

## 4.2 Classification of RNA-seq data

To classify RNA-seq gene expression profiles, we use the function `classifyProfile.rnaseq()`. It expects a reference and a test matrix. As a reference matrix we use an RNA-seq dataset derived from the study E-MTAB-1733 [7], which is available from ArrayExpress [8]. This dataset is named `se_rnaseq_refmat` and can be loaded with the following code:

```r
> library(sampleClassifierData)
> data("se_rnaseq_refmat")
> rnaseq_refmat <- assay(se_rnaseq_refmat)
> dim(rnaseq_refmat)
[1] 43819   71
```
As a test matrix we use an RNA-seq dataset derived from the study E-MTAB-513 [9], which is available from ArrayExpress. This dataset is named `se_rnaseq_testmat` and can be loaded with the following code:

```r
> data("se_rnaseq_testmat")
> rnaseq_testmat <- assay(se_rnaseq_testmat)
> dim(rnaseq_testmat)
[1] 43819 12
```

Now, we can call the function `classifyProfile.rnaseq()` to predict the classes of the samples in the test matrix:

```r
> res2.list <- classifyProfile.rnaseq(ref_matrix=rnaseq_refmat, query_mat=rnaseq_testmat, + gene.ids.type="ensembl",write2File=FALSE)

Detecting marker genes...
Done!
12 profiles to be classified...
Done!
```

For simplicity, we show only the two top hits for each query sample:

```r
> lapply(res2.list,"[",c(1,2),,drop=FALSE)

$adrenal
   Hits Score Ratio
 1  testis 0.141  26 / 184
 2  spleen 0.121  12 /  99

$brain
   Hits Score Ratio
 1   brain 0.75  138 / 184
 2 appendix 0.143  1 /  7

$colon
   Hits Score Ratio
 1 appendix 0.286  2 /  7
 2 endometrium 0.135  5 / 37

$heart
   Hits Score Ratio
 1   heart 0.543 100 / 184
 2     fat 0.103  19 / 184

$kidney
   Hits Score Ratio
 1  kidney 0.467  85 / 182
 2     lung 0.091  7 /  77

$liver
   Hits Score Ratio
 1   liver 0.652 120 / 184
 2 gallbladder 0.174  4 /  23
```
### $\text{lung}$

<table>
<thead>
<tr>
<th>Hits Score</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>lung</td>
<td>0.325</td>
</tr>
<tr>
<td>bonemarrow</td>
<td>0.201</td>
</tr>
</tbody>
</table>

### $\text{lymph.node}$

<table>
<thead>
<tr>
<th>Hits Score</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>lymphnode</td>
<td>0.196</td>
</tr>
<tr>
<td>appendix</td>
<td>0.143</td>
</tr>
</tbody>
</table>

### $\text{ovary}$

<table>
<thead>
<tr>
<th>Hits Score</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>ovary</td>
<td>0.196</td>
</tr>
<tr>
<td>endometrium</td>
<td>0.108</td>
</tr>
</tbody>
</table>

### $\text{prostate}$

<table>
<thead>
<tr>
<th>Hits Score</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>endometrium</td>
<td>0.243</td>
</tr>
<tr>
<td>colon</td>
<td>0.091</td>
</tr>
</tbody>
</table>

### $\text{testis}$

<table>
<thead>
<tr>
<th>Hits Score</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>testis</td>
<td>0.989</td>
</tr>
<tr>
<td>heart</td>
<td>0.092</td>
</tr>
</tbody>
</table>

### $\text{thyroid}$

<table>
<thead>
<tr>
<th>Hits Score</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>thyroid</td>
<td>0.255</td>
</tr>
<tr>
<td>heart</td>
<td>0.239</td>
</tr>
</tbody>
</table>

To display the classification results as a heatmap, we call the function `get.heatmap()` with the resulted list as input.

```r
> get.heatmap(res2.list)
```
In order to compare the classification predictions for RNA-seq query profiles to those of SVM, the function `classifyProfile.rnaseq.svm()` was implemented based on functions from the R-package `e1071`.

```r
> res2.svm.df <- classifyProfile.rnaseq.svm(ref_matrix=rnaseq_refmat, query_mat=rnaseq_testmat, + gene.ids.type="ensembl")

Detecting marker genes...
Done!
building an SVM model...
12 profiles to be classified...
done!

> res2.svm.df

query_name  predicted_class
1  adrenal       appendix
2    brain        brain
3    colon  gallbladder
4   heart         heart
5  kidney        kidney
```
Our method `classifyProfile.rnaseq()` classified 9 samples correctly, whereas SVM classified 7 of the 12 query samples correctly. To show the query samples that were misclassified by our method or SVM, we run the following code:

```r
> misclas.inds <- which(as.character(res2.svm.df[,1])!=as.character(res2.svm.df[,2]))
> colnames(res2.svm.df) <- c("query_real_class","predicted_class_by_SVM")
> pred.classifyProfile.rnaseq <- as.character(unlist(lapply(res2.list[which(names(res2.list) %in% 
+ as.character(res2.svm.df[misclas.inds,1])]),"[",1,1,drop=TRUE)))
> comp.df <- cbind(res2.svm.df[misclas.inds,],
+ predicted_by_classifyProfile.rnaseq=pred.classifyProfile.rnaseq)
> comp.df

query_real_class predicted_class_by_SVM predicted_by_classifyProfile.rnaseq
 1 adrenal         appendix            testis
 3 colon           gallbladder          appendix
 8 lymph.node      appendix            lymphnode
 9 ovary           endometrium          ovary
10 prostate       endometrium          endometrium
```

## 5 sampleClassifier algorithm details

The algorithm used in sampleClassifier is a simple algorithm called Shared Marker Genes (SMG). As the name suggests, the number of shared marker genes between a query and a reference is used as similarity measure. The tool requires a reference matrix with at least three replicates for each sample type. This matrix is used for marker gene detection using MGFM or MGFR. Since the number of detected markers differs depending on the sample types, we filter the list of marker genes of each sample type. Using the complete list of markers of each sample type, will result in a bias towards the sample type with the most marker genes. For example, if testis is the tissue with the most marker genes, using all marker genes for classification will result in classifying query samples often as testis. Suppose the reference matrix contains four tissues: liver, lung, kidney and midbrain, and \( X = (16, 20, 100, 500) \) is the vector of lengths of predicted marker genes for these tissues. The filtering is based on the median number of marker genes, in this case \( \text{median}(X) = 60 \). If the number of predicted markers for a tissue > 60, then only the top 60 marker genes will be used for classification. If the number of predicted markers for a tissue <= 60, then all markers are used for classification. After the filtering step, each query sample will be compared to all sample types in the reference and the number of marker genes shared between the query and each sample type in the reference is calculated. A query shares a marker gene with a reference sample if this marker gene is highly expressed in the query sample compared to all other sample types in the reference. The ratio of the number of shared marker genes and the total number of markers used for classification is used as a similarity score. This score has a value between 0 and 1. A value of 1 means that the query shares all marker genes with the
reference, and a value of 0 means that no marker genes are shared between the query and
the reference. For each query, the hits are sorted according to this score. The class of the
first top hit is predicted as a class for the query.

6 R sessionInfo

The results in this file were generated using the following packages:

> sessionInfo()

R version 4.4.0 beta (2024-04-15 r86425)
Platform: x86_64-pc-linux-gnu
Running under: Ubuntu 22.04.4 LTS

Matrix products: default
BLAS: /home/biocbuild/bbs-3.19-bioc/R/lib/libRblas.so
LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.10.0

locale:
[1] LC_CTYPE=en_US.UTF-8 LC_NUMERIC=C
[3] LC_TIME=en_US.UTF-8 LC_COLLATE=en_US.UTF-8
[5] LC_MONETARY=en_US.UTF-8 LC_MESSAGES=en_US.UTF-8
[7] LC_PAPER=en_US.UTF-8 LC_NAME=C
[9] LC_ADDRESS=C LC_TELEPHONE=C

time zone: America/New_York
tzcode source: system (glibc)

attached base packages:
[1] stats4 stats graphics grDevices utils datasets methods
[8] base

other attached packages:
[1] hgu133a.db_3.13.0 hgu133plus2.db_3.13.0
[3] org.Hs.eg.db_3.19.1 sampleClassifierData_1.27.0
[5] SummarizedExperiment_1.34.0 GenomicRanges_1.56.0
[7] GenomeInfoDb_1.40.0 MatrixGenerics_1.16.0
[9] matrixStats_1.13.0 sampleClassifier_1.28.0
[11] MGFR_1.30.0 MGF.M_1.38.0
[13] annotate_1.82.0 XML_3.99-0.16.1
[15] AnnotationDbi_1.66.0 IRanges_2.38.0
[17] S4Vectors_0.42.0 Biobase_2.64.0
[19] BiocGenerics_0.58.0

loaded via a namespace (and not attached):
[1] tidyselect_1.2.1 farver_2.1.1 dplyr_1.1.4
[4] blob_1.2.4 filelock_1.0.3 Biostrings_2.72.0
[7] fastmap_1.1.1 BioCFileCache_2.12.0 digest_0.6.35
[10] Lifecycle_1.0.4 KEGGREST_1.44.0 RSQLite_2.3.6
[13] magrittr_2.0.3 compiler_4.4.0 rlang_1.1.3
References


---

<table>
<thead>
<tr>
<th>Package</th>
<th>Version</th>
<th>Encoding</th>
<th>String Handling</th>
<th>DateTime</th>
<th>Table Handling</th>
<th>HTML Tools</th>
<th>Markup</th>
<th>XML</th>
<th>Bioc Mart</th>
<th>Genomic Data</th>
<th>Database</th>
<th>Date Handling</th>
<th>General Headers</th>
<th>Automation</th>
<th>Build System</th>
<th>Build System</th>
<th>Build System</th>
</tr>
</thead>
<tbody>
<tr>
<td>progress_1.2.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>yaml_2.3.8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S4Arrays_1.4.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DelayedArray_0.30.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>withr_3.0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xtable_1.8-4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ggplot2_3.3.5.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cli_3.6.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>generics_0.1.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cachem_1.0.8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>zlibbioc_1.50.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>vctrs_0.6.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>hms_1.1.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>stringi_1.8.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>munsell_0.5.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rappdirs_0.3.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R6_2.5.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>evaluate_0.23</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>memoise_2.0.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SparseArray_1.4.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Vignette for the *sampleClassifier* R Package

[16] progress_1.2.3 tools_4.4.0 utf8_1.2.4
[19] yaml_2.3.8 knitr_1.46 prettyunits_1.2.0
[22] S4Arrays_1.4.0 bit_4.0.5 curl_5.2.1
[25] DelayedArray_0.30.0 xml_2.1.3.6 abind_1.4-5
[28] withr_3.0.0 grid_4.4.0 fansi_1.0.6
[31] xtable_1.8-4 e1071_1.7-14 colorspace_2.1-0
[34] ggplot2_3.3.5.1 scales_1.3.0 biomat_2.60.0
[37] cli_3.6.2 rmarkdown_2.26 crayon_1.5.2
[40] generics_0.1.3 http_1.4.7 DBI_1.2.2
[43] cachem_1.0.8 proxy_0.4.27 stringr_1.5.1
[46] zlibbioc_1.50.0 BiocManager_1.30.22 XVector_0.4.4.0
[49] vctrs_0.6.5 Matrix_1.7-0 jsonlite_1.8.8
[52] hms_1.1.3 bit64_4.0.5 glue_1.7.0
[55] stringi_1.8.3 tibble_3.2.1 pillar_1.9.0
[58] munsell_0.5.1 htmltools_0.5.8.1 GenomeInfoDbData_1.2.12
[61] rappdirs_0.3.3 dbplyr_2.5.0 htr_1.0.1
[64] evaluate_0.23 lattice_0.22-6 png_0.1-8
[67] memoise_2.0.1 BiocStyle_2.32.0 class_7.3-22
[70] SparseArray_1.4.0 xfun_0.43 pkgconfig_2.0.3
Vignette for the sampleClassifier R Package


