Working with Multilabel Datasets in R: The **mldr** Package

Francisco Charte
F. David Charte

November 3, 2024

Abstract

Most classification algorithms deal with datasets which have a set of input features, the variables to be used as predictors, and only one output class, the variable to be predicted. However, in late years many scenarios in which the classifier has to work with several outputs have come to life. Automatic labeling of text documents, image annotation or protein classification are among them. Multilabel datasets are the product of these new needs, and they have many specific traits. The **mldr** package allows the user to load datasets of this kind, obtain their characteristics, produce specialized plots, and manipulate them. The goal is to provide the exploratory tools needed to analyze multilabel datasets, as well as the transformation and manipulation functions that will make possible to apply binary and multiclass classification models to this data or the development of new multilabel classifiers. Thanks to its integrated user interface, the exploratory functions will be available even to non-specialized R users.

1 Introduction

Pattern classification is an important task nowadays and is in use everywhere, from our e-mail client, which is able to separate spam from legit messages, to credit institutions, that rely on it to detect fraud and grant or deny loans. All these cases operate with binary datasets, since a message is either spam or legit, and multiclass datasets, the loan is safe, medium, risky or highly risky, for instance. In both cases the user expects only one output.

The huge growth on the amount of information stored in late years onto the Web, such as blog posts, pictures taken from cameras and phones, videos hosted on Youtube, and messages on social networks, has demanded a more complex classification work. A blog post can be classified into several non-exclusive categories, for instance news, economy and politics simultaneously. A picture can be assigned a set of labels, such as landscape, sky and forest. A video can be labeled into several musical styles at once, etc. All of these are examples of tasks in need of multilabel classification.

Binary and multiclass datasets can be managed in R by using dataframes. Usually the last attribute (column on the `data.frame`) is the output class, whether it contains only `TRUE/FALSE` values or a value belonging to a finite set (a factor). Multilabel datasets (MLDs) can also be stored in an R `data.frame`, but an additional structure to know which attributes are output labels is needed. Moreover, this kind of datasets have many specific characteristics that do not exist in the traditional ones. The average number of labels per instance, the imbalance ratio for each label, the number of labelsets (sets of labels assigned to each row) and their frequencies, and the level of concurrence among imbalanced labels are some of the traits that differentiate an MLD from the others.

Until now, most of the software to work with MLDs has been written in Java. The two best known frameworks are MULAN [19] and MEKA [13]. Both rely on WEKA, since it offers a large variety of binary and multiclass classifiers, as well as the functions needed to deal with ARFF (*Attribute-Relation File Format*) files. Most of the existent MLDs are stored in ARFF format. MULAN and MEKA provide the specialized tools needed to deal with multilabel ARFFs, and the infrastructure to build multilabel classifiers (MLCs). Although R can access WEKA services through the **RWeka** [10] package, handling MLDs is far from an easy task. This has been the main motivation behind the **mldr** package development. In the best of our knowledge, **mldr** is the first R package aimed to ease the work with multilabel data.

The **mldr** package aims to provide the user with the functions needed to perform exploratory analysis over MLDs, stating their main traits both statistically and visually. Moreover, it also brings the proper tools to manipulate this kind of datasets, including the application of the most common transformation methods, BR (*Binary Relevance*) and LP (*Label Powerset*), that will be described in the following section. These would be the foundation for processing the MLDs with traditional classifiers, as well as for developing new multilabel algorithms.

The **mldr** package does not depend on the **RWeka** package and it is not linked to MULAN nor MEKA. It has been designed to allow reading both MULAN and MEKA MLDs, but without any external dependencies. In fact, it would be possible to load MLDs stored in other file formats, as well as creating them from scratch. The package will create for the user an `"mldr"` object, containing the data in the MLD and also a large set of measures obtained from it. The functions provided by the package ease the access to this information, produce some specific plots, and make possible the manipulation of its content. A web-based user interface, developed using the **shiny** [2] package, puts the exploratory analysis tools of the **mldr** package at the fingertips of all users, even those which have little experience using R.

In the following section the foundations related to MLDs and MLC will be briefly introduced. After that, the structure of the **mldr** package and the operations it provides will be explained. Finally, the user interface provided by **mldr** to ease exploratory analysis tasks over MLDs will be shown. All code displayed in this paper is available in a vignette, accessible by loading the **mldr** package and entering `vignette("mldr")`.

2 Working with multilabel datasets

MLDs are generated from text documents [11], sets of images [6], music collections, and protein attributes [5], among other sources. For each sample a set of features (input attributes) is collected, and a set of labels (the output labelset) is assigned. Usually there are several hundreds or even thousands of attributes, and it is not rare that a MLD has more labels than features. Some MLDs have only a few labels per instance, while others have dozens of them. In some MLDs the number of label combinations (labelsets) is quite short, whereas in others it can be very large. Most MLDs are imbalanced, which means that some labels are very frequent while others are scarcely represented. The labels in an MLD can be correlated or not. Moreover, frequent labels and rare labels can appear together in the same instances.

As can be seen, a lot of different scenarios can be found depending on the MLD characteristics. This is the reason why several specific measures have been designed to assess MLD traits [18], since they can have a serious impact into the MLCs performance. The two subsections below introduce several of these measures and some of the approaches followed to face multilabel classification.

2.1 Multilabel dataset traits

The most common characterization measures for MLDs can be grouped into four categories, as depicted in Figure 1.

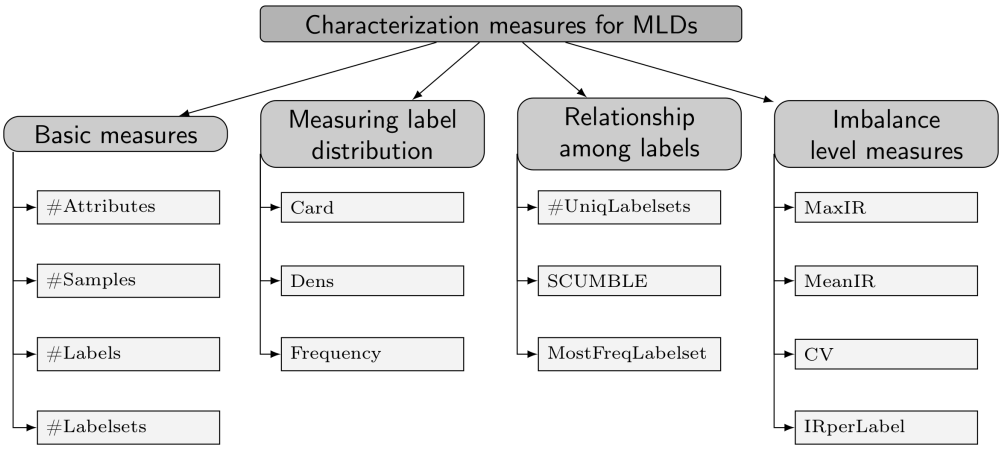


Figure 1: Characterization measures taxonomy.

The most basic information that can be obtained from an MLD is the number of instances, attributes and labels. Being D any MLD containing $|D|$ instances, any instance $D_i, i \in \{1..|D|\}$ will be the union of a set of attributes and a set of labels (X_i, Y_i), $X_i \in X^1 \times X^2 \times \cdots \times X^f, Y_i \subseteq L$, where f is the number of input features and X^j is the space of possible values for the j-th attribute, $j \in \{1..f\}$. L being the full set of labels used in D, Y_i could be any subset of items in L. Therefore, theoretically the number of potential labelsets could be $2^{|L|}$. In practice this number tends to be limited by $|D|$.

Each instance D_i has an associated labelset, whose length (number of active labels) can be in the range $\{0..|L|\}$. The average number of active labels per instance is the most basic measure of any MLD, usually known as *Card* (standing for cardinality). It is calculated as shown in Equation 1. Dividing this measure by the number of labels in L, as shown in Equation 2, results in a dimension-less measure, known as *Dens* (standing for label density).

$$Card(D) = \frac{1}{|D|}\sum_{i=1}^{|D|}|Y_i|. \tag{1}$$

$$Dens(D) = \frac{1}{|D|}\sum_{i=1}^{|D|}\frac{|Y_i|}{|L|}. \tag{2}$$

Most multilabel datasets are imbalanced, meaning that some of the labels are very frequent whereas others are quite rare. The level of imbalance of a determinate label can be measured by the imbalance ratio, $IRLbl$, defined in Equation 3. To know how much imbalance there is in D, the $MeanIR$ measure [4] is calculated as the mean imbalance ratio among all labels, as shown in Equation 4. In order to know the significance of this last measure, the standard CV (*Coefficient of Variation, Equation 5*) can be used.

$$IRLbl(y) = \frac{\underset{y' \in L}{\operatorname{argmax}} \left(\sum_{i=1}^{|D|} h(y', Y_i) \right)}{\sum_{i=1}^{|D|} h(y, Y_i)}, \quad h(y, Y_i) = \begin{cases} 1 & y \in Y_i \\ 0 & y \notin Y_i \end{cases}. \tag{3}$$

$$MeanIR = \frac{1}{|L|} \sum_{y \in L} (IRLbl(y)). \tag{4}$$

$$CV = \frac{IRLbl\sigma}{MeanIR}, \; IRLbl\sigma = \sqrt{\sum_{y=Y_1}^{Y_{|Y|}} \frac{(IRLbl(y) \; - \; MeanIR)^2}{|Y| - 1}} \tag{5}$$

The number of different labelsets, as well as the amount of them being unique labelsets (appearing only once in D), give us a glimpse on how sparsely the labels are distributed. The labelsets by themselves allow to know how the labels in L are related. A very frequent labelset denotes that the labels in it tend to appear jointly in D. The $SCUMBLE$ measure, introduced in [3] and shown in Equation 7, is used to assess the concurrence level among frequent and infrequent labels.

$$SCUMBLE_{ins}(i) = 1 - \frac{1}{\overline{IRLbl_i}} (\prod_{l=1}^{|L|} IRLbl_{il})^{(1/|L|)} \tag{6}$$

$$SCUMBLE(D) = \frac{1}{|D|} \sum_{i=1}^{|D|} SCUMBLE_{ins}(i) \tag{7}$$

Besides the aforementioned, there are some other interesting traits that can be indirectly obtained from the previous measures, such as the ratio between input features and output labels, the maximum $IRLbl$, or the coefficient of variation in the imbalance levels, among others.

Although the raw numbers given by these calculations describe the nature of any multilabel dataset to a good level, in general a visualization of its characteristics would be desirable to ease its interpretation by researchers.

The information obtained from the previous measures depicts the characteristics of the dataset. This data, along with other factors such as the loss function used by the classifier, would help in choosing the most proper algorithm to learn from it and, in the future, make predictions on new data. Traditional classification models, such as trees and support vector machines, are designed to give only one output as result. Multilabel classification can mainly be faced through two different approaches:

2.2 Multilabel classification

- **Algorithm adaptation**: The goal is to modify existent algorithms taking into account the multilabel nature of the samples, for instance hosting more than one class in the leaves of a tree instead of only one.
- **Problem transformation**: This approach transforms the original data to make it suitable to traditional classification algorithms, then combines the obtained predictions to build the labelsets given as output result.

Although several transformation methods have been defined in the specialized literature, there are two among them that stand out because they are the foundation for many others:

- **Binary Relevance** (BR): Introduced by [8] as an adaptation of OVA (*one-vs-all*) to the multilabel scenario, this method transforms the original multilabel dataset into several binary datasets, as many as different labels there are. This way any binary classifier can be used, joining their individual predictions to generate the final output.
- **Label Powerset** (LP): Introduced by [1], this method transforms the multilabel dataset into a multiclass dataset by using the labelset of each instance as class identifier. Any multiclass classifier can be used, transforming back the predicted class into a labelset.

BR and LP have been used not only as a direct technique to implement multilabel classifiers, but also as base methods to build more sophisticated algorithms. Several ensembles of binary classifiers relying on BR have been proposed, such as CC (*Classifier Chains*) or ECC (*Ensemble of Classifier Chains*), both by [15]. The same is applicable to the LP transformation, foundation of ensemble multilabel classifiers such as RAkEL (Random k-Labelsets for Multi-Label Classification, [17]) and EPS (Ensemble of Pruned Sets, [14]).

For the readers interested in deeper knowledge, a recent review on multilabel classification has been published by [21].

3 The mldr package

R is among the most used tools when it comes to performing data mining tasks, including binary and multiclass classification. However, the work with MLDs in R is not as easy as it is with classic datasets. This is the main motivation behind the development of the **mldr** package, whose goals and functionality are described in this section.

3.1 Main goals of the mldr package

When we planned the development of this package, our main objective was to ease the exploration of MLDs from R. This included loading existent MLDs in different formats, as well as obtaining from them all the available information. These functions should be available to everyone, even to users not used to the R command line but to GUIs (*Graphic User Interfaces*) such as **Rcmdr** (aka *R Commander*, [7]) or **rattle** [20].

At the same time, we aimed to include the tools needed to manipulate the MLDs, apply filters and transformations, and also the possibility of loading MLDs from alternative file formats, as well as to creating them from scratch. This functionality, directed to more experienced R users, opens the doors to implement other algorithms on top of **mldr**, for instance preprocessing methods or multilabel classifiers.

3.2 Installing and loading the mldr package

The **mldr** package is available at CRAN servers, therefore it can be installed as any other package, by simply typing:

```
install.packages("mldr")
```

mldr depends on three R packages: **XML** [12], **circlize** [9] and **shiny**. The first one allows reading XML (*eXtensible Markup Language*) files, the second one is used to generate a specific type of plot (described below), and the third one is the base of its user interface.

Older releases of **mldr**, as well as the development version, are available at `http://github.com/fcharte/mldr`. It is possible to install the development version using the `install_github()` function from **devtools**.

Once installed, the package has to be loaded before it can be used. This can be done through the `library()` or `require()` functions, as usual. After loading the package three sample MLDs will be available: `birds`, `emotions` and `genbase`. These are included into the `'birds.rda'`, `'emotions.rda'` and `'genbase.rda'` files, which are lazily loaded along with the package.

The **mldr** package uses its own internal representation for MLDs, which are assigned the `"mldr"` class. Inside an `"mldr"` object, such as the previous mentioned `emotions` or `birds`, both the data in the MLD and all the information obtained from this data can be found.

3.3 Loading and creating MLDs

Besides the three sample MLDs included in the package, the `mldr()` function allows to load any MLD stored in MULAN or MEKA file formats. Assuming that the files `'corel5k.arff'` and `'corel5k.xml'`, which hold the Corel5k [6] MLD in MULAN format, are in the current directory, the loading will be done as follows:

```
corel5k <- mldr("corel5k")
```

If the XML file is not available, it is possible to indicate just the number of labels in the MLD instead. In this case, the function assumes that the labels are at the end of the list of features. For instance:

```
corel5k <- mldr("corel5k", label_amount = 374)
```

Loading an MLD in MEKA file format is equally easy. In this case there is not an XML file with label information, but a special header inside the ARFF file, a fact that will be indicated to `mldr()` with the `use_xml` parameter:

```
imdb <- mldr("imdb", use_xml = FALSE)
```

In all cases the result, as long as the MLD can be correctly loaded and parsed, will be a new `"mldr"` object ready to use.

If the MLD we are interested in is not in MULAN or MEKA format, firstly it will have to be loaded in a `data.frame`, for instance using functions such as `read.csv()`, `read.table()` or a more specialized reader, and secondly this `data.frame` and an integer vector stating the indices of the labels inside it will be given to the `mldr_from_dataframe()` function. This is a general function for creating an `"mldr"` object from any `data.frame`, so it can also be used to generate new MLDs on the fly, as shown in the following example:

```
df <- data.frame(matrix(rnorm(1000), ncol = 10))
df$Label1 <- c(sample(c(0,1), 100, replace = TRUE))
df$Label2 <- c(sample(c(0,1), 100, replace = TRUE))
mymldr <- mldr_from_dataframe(df, labelIndices = c(11, 12), name = "testMLDR")
```

This will assign to `mymldr` an MLD, named `testMLDR`, with 10 input attributes and 2 labels.

3.4 Obtaining information from an MLD

After loading any MLD, a quick summary of its main characteristics can be obtained by means of the usual `summary()` function, as shown below:

```
summary(birds)
```

```
##   num.attributes num.instances num.inputs num.labels num.labelsets
## 1            279           645        260         19           133
##   num.single.labelsets max.frequency cardinality    density   meanIR    scumble
## 1                   73           294    1.013953 0.05336597 5.406996 0.03302765
##   scumble.cv      tcs
## 1   2.298296 13.39547
```

Any of these measures can be individually obtained through the `measures` member of the `"mldr"` class, like this:

```
emotions$measures$num.attributes
```

```
## [1] 78
```

```
genbase$measures$scumble
```

```
## [1] 0.0287591
```

Full information about the labels in the MLD, including the number of times they appear, their *IRLbl* and *SCUMBLE* measures, can be retrieved by using the `labels` member of the `"mldr"` class:

```
birds$labels
```

```
##                           index count        freq     IRLbl    SCUMBLE
## Brown Creeper               261    14 0.021705426  7.357143 0.12484341
## Pacific Wren                262    81 0.125581395  1.271605 0.05232609
## Pacific-slope Flycatcher    263    46 0.071317829  2.239130 0.06361470
## Red-breasted Nuthatch       264     9 0.013953488 11.444444 0.15744451
## Dark-eyed Junco             265    20 0.031007752  5.150000 0.10248336
## Olive-sided Flycatcher      266    14 0.021705426  7.357143 0.18493760
## Hermit Thrush               267    47 0.072868217  2.191489 0.06777263
## Chestnut-backed Chickadee   268    40 0.062015504  2.575000 0.06807452
## Varied Thrush               269    61 0.094573643  1.688525 0.07940806
## Hermit Warbler              270    53 0.082170543  1.943396 0.07999006
## Swainson's Thrush           271   103 0.159689922  1.000000 0.11214301
## Hammond's Flycatcher        272    28 0.043410853  3.678571 0.06129884
## Western Tanager             273    33 0.051162791  3.121212 0.07273988
## Black-headed Grosbeak       274     9 0.013953488 11.444444 0.20916487
## Golden Crowned Kinglet      275    37 0.057364341  2.783784 0.09509474
## Warbling Vireo              276    17 0.026356589  6.058824 0.14333613
## MacGillivray's Warbler      277     6 0.009302326 17.166667 0.24337605
## Stellar's Jay               278    10 0.015503876 10.300000 0.12151527
## Common Nighthawk            279    26 0.040310078  3.961538 0.06520272
##                           SCUMBLE.CV
## Brown Creeper              0.6788629
## Pacific Wren               1.4590810
## Pacific-slope Flycatcher   1.1456585
## Red-breasted Nuthatch      0.9686548
## Dark-eyed Junco            0.7750301
## Olive-sided Flycatcher     0.5305291
## Hermit Thrush              1.2703174
## Chestnut-backed Chickadee  1.1641298
## Varied Thrush              1.1296842
## Hermit Warbler             1.4472129
## Swainson's Thrush          0.9562594
## Hammond's Flycatcher       0.9188832
## Western Tanager            0.8804317
## Black-headed Grosbeak      0.6308997
## Golden Crowned Kinglet     0.8898242
## Warbling Vireo             0.9121402
## MacGillivray's Warbler     0.8726136
## Stellar's Jay              1.5532108
## Common Nighthawk           1.4424929
```

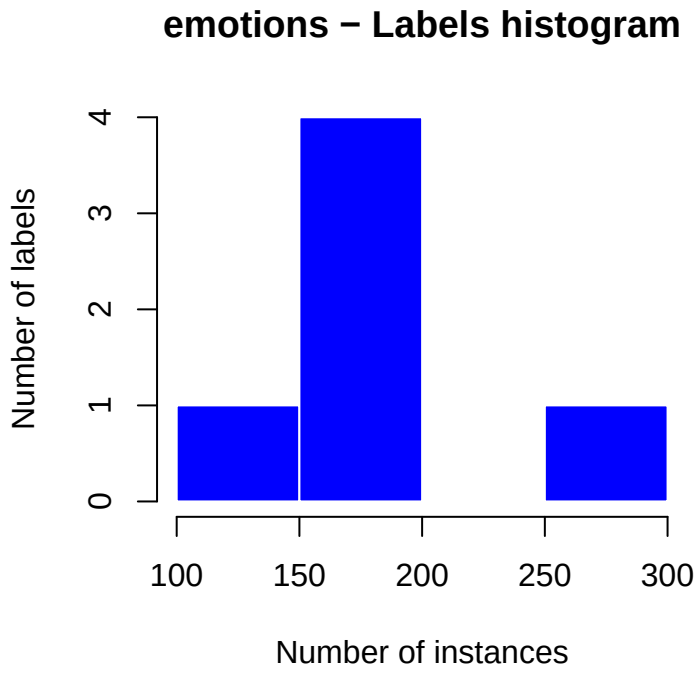
The same is applicable for labelsets and attributes, by means of the `labelsets` and `attributes` members of the class.

To access the MLD content, attributes and label values, the `print()` function can be used, as well as the `dataset` member of the `"mldr"` object.

3.5 Plotting functions

Exploratory analysis of MLDs can be tedious, since most of them have thousands of attributes and hundreds of labels. The **mldr** package provides a `plot()` function specific for dealing with `"mldr"` objects, allowing the generation of several specific types of plots. The first parameter given to `plot()` must be an `"mldr"` object, while the second one specifies the type of plot to be produced.

```
plot(emotions, type = "LH")
```



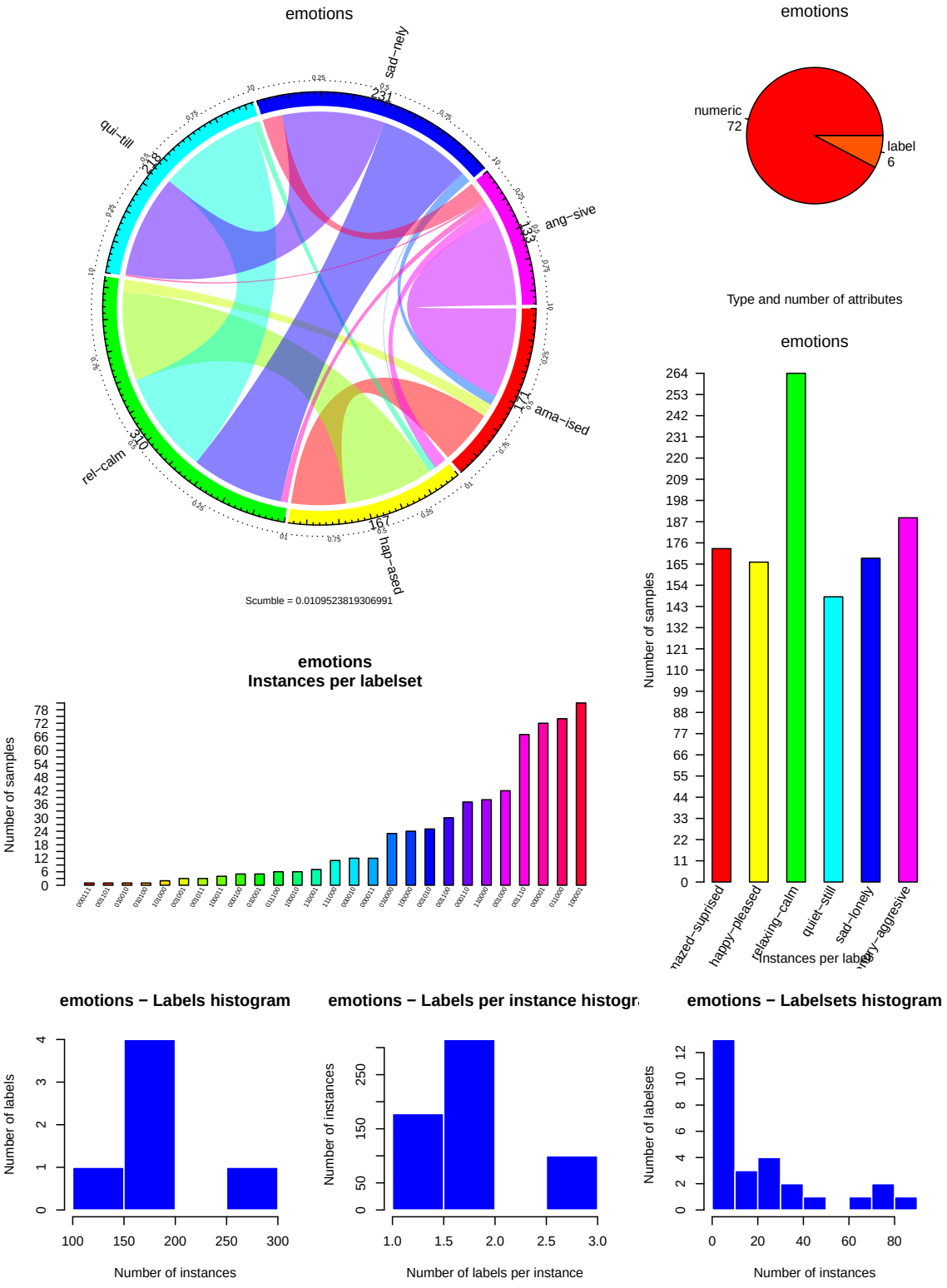
There are seven different types of plots available: three histograms showing relations between instances and labels, two bar plots with similar aim, a circular plot indicating types of attributes and a concurrence plot for labels. All of them are shown in the following code:

```
layout(matrix(c(1,1,6,1,1,3,5,5,3,2,4,7), 4, 3, byrow = TRUE))

plot(emotions, type = "LC")
```

```
## `major.tick.percentage` is not used any more, please directly use argument `major.tick.length`.
## `major.tick.percentage` is not used any more, please directly use argument `major.tick.length`.
## `major.tick.percentage` is not used any more, please directly use argument `major.tick.length`.
## `major.tick.percentage` is not used any more, please directly use argument `major.tick.length`.
## `major.tick.percentage` is not used any more, please directly use argument `major.tick.length`.
## `major.tick.percentage` is not used any more, please directly use argument `major.tick.length`.
```

```
plot(emotions, type = "LH")
plot(emotions, type = "LB")
plot(emotions, type = "CH")
plot(emotions, type = "LSB")
plot(emotions, type = "AT")
plot(emotions, type = "LSH")
```



The label histogram (type "LH") relates labels and instances in a way that shows how well-represented labels are in general. The X axis is assigned to the number of instances and the Y axis to the amount of labels. This means that if a great number of labels are appearing in very few instances, all data will concentrate on the left side of the plot. On the contrary, if labels are generally present in many instances, data will tend to accumulate on the right side. This plot shows imbalance of labels when there is data accumulated on both sides of the plot, which implies that many labels are underrepresented and a large amount are overrepresented as well.

The labelset histogram (named "LSH") is similar to the former in a way. However, instead of representing the number of instances in which each label appears, it shows the amount of labelsets. This indicates quantitatively whether labelsets repeat consistently or not among instances.

The label and labelset bar plots display exactly the number of instances for each one of the labels and labelsets, respectively. Their codes are "LB" for the label bar plot and "LSB" for the labelset one.

The cardinality histogram (type "CH") represents the amount of labels instances have in general, therefore

data accumulating on the right side of the plot will mean that instances do have a notable amount of labels, whereas data concentrating on the left side shows the opposite situation.

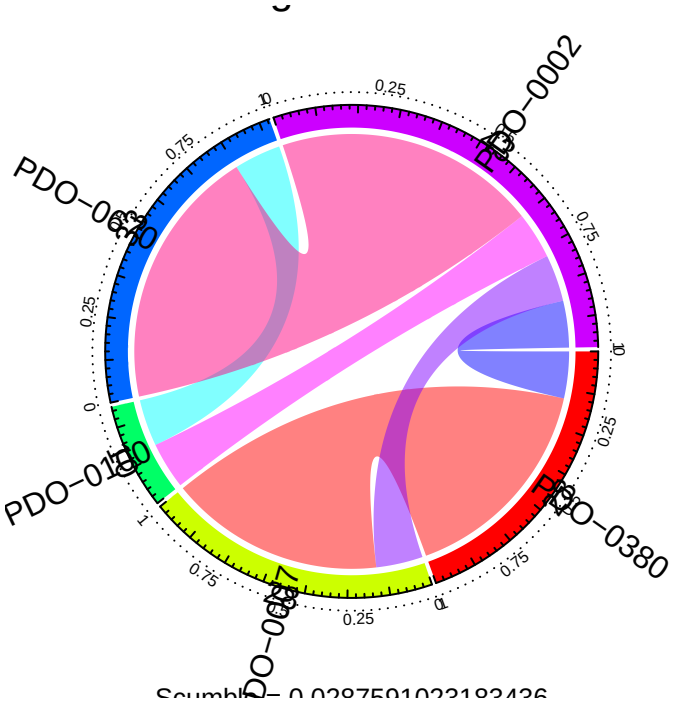
The attribute types plot (named `"AT"`) is a pie chart displaying the number of labels, numeric attributes and finite set (character) attributes, thus showing the proportions between these types of attributes to ease the comprehension about the amount of input information and that of output data.

The concurrence plot is the default one, with type `"LC"`, and responds to the need of exploring interactions among labels, and specifically between majority and minority ones. This plot has a circular shape, with the circumference partitioned into several disjoint arcs representing labels. Each arc has length proportional to the number of instances where the label is present. These arcs are in turn divided into bands that join two of them, showing the relation between the corresponding labels. The width of each band indicates the strength of the relation, since it is proportional to the number of instances in which both labels appear simultaneously. In this manner, a concurrence plot can show whether imbalanced labels appear frequently together, a situation which could limit the possible improvement of a preprocessing technique [3].

Since drawing interactions among lots of labels can produce a confusing result, this last type of plot accepts more parameters: `labelCount`, which accepts an integer that will be used to generate the plot with that number of labels chosen at random; and `labelIndices`, which allows to indicate exactly the indices of the labels to be displayed in the plot. For example, in order to plot relations among the first eleven labels of `genbase`:

```
plot(genbase, labelIndices = genbase$labels$index[1:11])
```

```
## 'major.tick.percentage' is not used any more, please directly use argument 'major.tick.length'.
## 'major.tick.percentage' is not used any more, please directly use argument 'major.tick.length'.
## 'major.tick.percentage' is not used any more, please directly use argument 'major.tick.length'.
## 'major.tick.percentage' is not used any more, please directly use argument 'major.tick.length'.
## 'major.tick.percentage' is not used any more, please directly use argument 'major.tick.length'.
```



3.6 Transforming and filtering functions

Manipulation of datasets is a crucial task in multilabel classification. Since transformation is one of the main approaches to tackle the problem, both BR and LP transformations are implemented in **mldr**. They can be obtained by means of the `mldr_transform` function, which accepts an `"mldr"` object as first parameter, the type of transformation, `"BR"` or `"LP"`, as second, and an optional vector of label indices to be included in the transformation as last argument:

```
emotionsbr <- mldr_transform(emotions, type = "BR")

emotionslp <- mldr_transform(emotions, type = "LP", emotions$labels$index[1:4])
```

The BR transformation will return a list of `data.frame` objects, each one of them using one of the labels as class, whereas the LP transformation will return a single `data.frame` representing a multiclass dataset using each labelset as a class. Both of these transformations can be directly used in order to apply binary and multiclass classification algorithms, or even implement new ones.

```
emo_lp <- mldr_transform(emotions, "LP")
library(RWeka)
classifier <- IBk(classLabel ~ ., data = emo_lp, control = Weka_control(K = 10))
evaluate_Weka_classifier(classifier, numFolds = 5)
```

A filtering utility is included in the package as well. Using it is intuitive, since it can be called with the square bracket operator `[`. This allows to partition an MLD or filter it according to a logical condition.

```
emotions$measures$num.instances
```

```
## [1] 593
```

```
emotions[emotions$dataset$.SCUMBLE > 0.01]$measures$num.instances
```

```
## [1] 222
```

Combined with the joining operator, `+`, this can enable users to implement new preprocessing techniques that modify information in the MLD in order to improve classification results. For example, the following would be an implementation of an algorithm disabling majority labels on instances with highly imbalanced labels:

```
mldbase <- mld[.SCUMBLE <= mld$measures$scumble]

# Samples with coocurrence of highly imbalanced labels
mldhigh <- mld[.SCUMBLE > mld$measures$scumble]
majIndexes <- mld$labels[mld$labels$IRLbl < mld$measures$meanIR,"index"]

# Deactivate majority labels
mldhigh$dataset[, majIndexes] <- 0
joined <- mldbase + mldhigh # Join the instances without changes with the filtered ones
```

In this last example, the first two commands filter the MLD, separating instances with their *SCUMBLE* lower than the mean and those with it higher. Then, the third line obtains the indices of the labels with lower *IRLbl* than the mean, thus these are the majority labels of the dataset. Finally, these labels are set to 0 in the instances with high *SCUMBLE* and then the two partitions are joined again.

Lastly, another useful feature included in the **mldr** package is the MLD comparison with the `==` operator. This indicates whether both MLDs in comparison share the same structure, which would mean they have the same attributes and these would have the same type.

```
emotions[1:10] == emotions[20:30]
```

```
## [1] TRUE
```

```
emotions == birds
```

```
## [1] FALSE
```

3.7 Assessing multilabel predictive performance

Assuming that a set of predictions has been obtained for a MLD, through a set of binary classifiers, a multiclass classifier or any other algorithm, the next step would be to evaluate the classification performance. In the literature there exist more than 20 metrics for this task, and some of them are quite complex to calculate. The **mldr** package provides the `mldr_evaluate` function to accomplish this task, supplying both example based and label based metrics.

Multilabel evaluation metrics are grouped into two main categories: example based and label based metrics. Example based metrics are computed individually for each instance, then averaged to obtain the final value. Label based metrics are computed per label, instead of per instance. There are two approaches called *micro-averaging* and *macro-averaging* (described below). The output of the classifier can be a bipartition (i.e. a set of 0s and 1s denoting the predicted labels) or a ranking (i.e. a set of real values denoting the relevance of each label). For this reason, there are bipartition based and ranking based evaluation metrics for each one of the two previous categories.

D being the MLD, L the full set of labels used in D, Y_i the subset of predicted labels for the i-*th* instance, and Z_i the true subset of labels, the example/bipartition based metrics returned by `mldr_evaluate` are the following:

- **Accuracy**: It is defined (see Equation 8) as the proportion of correctly predicted labels with respect to the total number of labels for each instance.

$$Accuracy = \frac{1}{|D|}\sum_{i=1}^{|D|}\frac{|Y_i \cap Z_i|}{|Y_i \cup Z_i|} \tag{8}$$

- **Precision**: This metric is computed as indicated in Equation 9, giving as result the ratio of relevant labels predicted by the classifier.

$$Precision = \frac{1}{|D|}\sum_{i=1}^{|D|}\frac{|Y_i \cap Z_i|}{|Z_i|} \tag{9}$$

- **Recall**: It is a metric (see Equation 10) commonly used along with the previous one, measuring the proportion of predicted labels which are relevant.

$$Recall = \frac{1}{|D|}\sum_{i=1}^{|D|}\frac{|Y_i \cap Z_i|}{|Y_i|} \tag{10}$$

- **F-measure**: As can be seen in Equation 11, this metric is the harmonic mean between *Precision* (see Equation 9) and *Recall* (see Equation 10), providing a balanced assessment between precision and sensitivity.

$$F\text{-}Measure = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{11}$$

- **Hamming Loss**: It is the most common evaluation metric in multilabel literature, computed (see Equation 12) as the simmetric difference between predicted and true labels and divided by the total number of labels in the MLD.

$$HammingLoss = \frac{1}{|D|}\sum_{i=1}^{|D|}\frac{|Y_i \triangle Z_i|}{|L|} \tag{12}$$

- **Subset Accuracy**: This metric is also known as *0/1 Subset Accuracy* and *Classification Accuracy*, and it is the most strict evaluation metric. The $[\![expr]\!]$ operator (see Equation 13) returns 1 when *expr* is truthy and 0 otherwise, in this case its value is 1 only if the predicted set of labels equals to the true one.

$$SubsetAccuracy = \frac{1}{|D|}\sum_{i=1}^{|D|}[\![Y_i = Z_i]\!] \tag{13}$$

Let $rank(x_i, y)$ be a function returning the position of y, a certain label, in the x_i instance. The example/ranking based evaluation metrics returned by the `mldr_evaluate` function are the following ones:

- **Average Precision**: This metric (see Equation 14) computes the proportion of labels ranked ahead of a certain relevant label. The goal is to establish how many positions have to be traversed until this label is found.

$$AveragePrecision = \frac{1}{|D|}\sum_{i=1}^{|D|}\frac{1}{|Y_i|}\sum_{y \in Y_i}\frac{|\{y' \in Y_i : rank(x_i, y') \leq rank(x_i, y)\}|}{rank(x_i, y)} \tag{14}$$

- **Coverage**: Defined as indicated in Equation 15, this metric calculates the extent to which it is necessary to go up in the ranking to cover all relevant labels.

$$Coverage = \frac{1}{|D|}\sum_{i=1}^{|D|}\operatorname{argmax}_{y \in Y_i}\langle rank(x_i, y)\rangle - 1 \tag{15}$$

- **One Error**: It is a metric (see Equation 16) which determines how many times the best ranked label given by the classifier is not part of the true label set of the instance.

$$OneError = \frac{1}{|D|}\sum_{i=1}^{|D|}\left[\!\!\left[\operatorname{argmax}_{y \in Z_i}\langle rank(x_i, y)\rangle \notin Y_i\right]\!\!\right] \tag{16}$$

- **Ranking Loss**: This metric (see Equation 17) compares each pair of labels in L, computing how many times a relevant label (member of the true labelset) appears ranked lower than a non-relevant label. In the equation, $\overline{Y_i}$ is notation for $L \backslash Y_i$.

$$RankingLoss = \frac{1}{|D|}\sum_{i=1}^{|D|}\frac{1}{|Y_i||\overline{Y_i}|}\left|\left\{(y_a, y_b) \in Y_i \times \overline{Y_i} : rank(x_i, y_a) > rank(x_i, y_b)\right\}\right| \tag{17}$$

Regarding the label based metrics, there are two different ways to aggregate the values of the labels. The macro-averaging approach (see Equation 18) computes the metric independently for each label, then averages the obtained values to get the final measure. On the contrary, the micro-averaging approach (see Equation 19) first agregate the counters for all the labels, the computes the metric only once. In these equations *TP*, *FP*, *TN* and *FN* stand for *True Positives*, *False Positives*, *True Negatives* and *False Negatives*, respectively.

$$MacroMetric = \frac{1}{|L|}\sum_{l=1}^{|L|}evalMetric(TP_l, FP_l, TN_l, FN_l) \tag{18}$$

$$MicroMetric = evalMetric(\sum_{l=1}^{|L|} TP_l, \sum_{l=1}^{|L|} FP_l, \sum_{l=1}^{|L|} TN_l, \sum_{l=1}^{|L|} FN_l) \tag{19}$$

All the bipartition based metrics, such as *Precision*, *Recall* or *FMeasure*, can be computed as label based measures following these two approaches. In this category, there are as well some ranking based metrics, such as *MacroAUC* (see Equation 20) and *MicroAUC* (Equation 21).

$$MacroAUC = \frac{1}{|L|} \sum_{l=1}^{|L|} \frac{|\{x', x'' : rank(x', y_l) \geq rank(x'', y_l), (x', x'') \in X_l \times \overline{X_l}\}|}{|X_l||\overline{X_l}|}, \quad X_l = \{x_i : y_l \in Y_i\}, \overline{X_l} = \{x_i : y_l \notin Y_i\} \tag{20}$$

$$MicroAUC = \frac{|\{x', x'', y', y'' : rank(x', y') \geq rank(x'', y''), (x', y') \in S^+, (x'', y'') \in S^-\}|}{|S^+||S^-|}, \quad S^+ = \{(x_i, y) : y \in Y_i\}, S^- = \{(x_i, y) : y \notin Y_i\} \tag{21}$$

When the partition of the MLD for which the predictions have been obtained and the predictions themselves are given to the `mldr_evaluate` function, a list of 20 measures is returned. For instance:

```
# Get the true labels in emotions
predictions <- as.matrix(emotions$dataset[,emotions$labels$index])
# and introduce some noise
predictions[sample(1:593, 100),sample(1:6, 100, replace = TRUE)] <- sample(0:1, 100,
  replace = TRUE)
# then evaluate the predictive performance
res <- mldr_evaluate(emotions, predictions)

## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
```

```
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
```

```
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
```

```
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
## Setting levels:  control = 0, case = 1
## Setting direction:  controls < cases
```

```
str(res)
```

```
## List of 20
##  $ accuracy         : num 0.861
##  $ example_auc      : num 0.916
##  $ average_precision: num 0.924
##  $ coverage         : num 1.26
##  $ fmeasure         : num 0.875
##  $ hamming_loss     : num 0.0897
##  $ macro_auc        : num 0.916
##  $ macro_fmeasure   : num 0.862
##  $ macro_precision  : num 0.806
##  $ macro_recall     : num 0.928
##  $ micro_auc        : num 0.915
##  $ micro_fmeasure   : num 0.866
##  $ micro_precision  : num 0.812
##  $ micro_recall     : num 0.927
##  $ one_error        : num 0.126
##  $ precision        : num 0.861
##  $ ranking_loss     : num 0.169
##  $ recall           : num 0.927
##  $ subset_accuracy  : num 0.831
##  $ roc              :List of 15
##   ..$ percent           : logi FALSE
##   ..$ sensitivities     : num [1:3] 1 0.927 0
##   ..$ specificities     : num [1:3] 0 0.903 1
##   ..$ thresholds        : num [1:3] -Inf 0.5 Inf
##   ..$ direction         : chr "<"
##   ..$ cases             : num [1:1108] 1 1 1 1 0 1 1 1 1 1 ...
##   ..$ controls          : num [1:2450] 0 0 0 0 0 0 1 1 0 0 ...
##   ..$ fun.sesp          :function (thresholds, controls, cases, direction)
##   ..$ auc               : 'auc' num 0.915
##   .. ..- attr(*, "partial.auc")= logi FALSE
##   .. ..- attr(*, "percent")= logi FALSE
##   .. ..- attr(*, "roc")=List of 15
##   .. .. ..$ percent           : logi FALSE
##   .. .. ..$ sensitivities     : num [1:3] 1 0.927 0
##   .. .. ..$ specificities     : num [1:3] 0 0.903 1
##   .. .. ..$ thresholds        : num [1:3] -Inf 0.5 Inf
##   .. .. ..$ direction         : chr "<"
##   .. .. ..$ cases             : num [1:1108] 1 1 1 1 0 1 1 1 1 1 ...
##   .. .. ..$ controls          : num [1:2450] 0 0 0 0 0 0 1 1 0 0 ...
##   .. .. ..$ fun.sesp          :function (thresholds, controls, cases, direction)
##   .. .. ..$ auc               : 'auc' num 0.915
##   .. .. .. ..- attr(*, "partial.auc")= logi FALSE
##   .. .. .. ..- attr(*, "percent")= logi FALSE
##   .. .. .. ..- attr(*, "roc")=List of 8
##   .. .. .. .. ..$ percent      : logi FALSE
##   .. .. .. .. ..$ sensitivities: num [1:3] 1 0.927 0
##   .. .. .. .. ..$ specificities: num [1:3] 0 0.903 1
##   .. .. .. .. ..$ thresholds   : num [1:3] -Inf 0.5 Inf
##   .. .. .. .. ..$ direction    : chr "<"
##   .. .. .. .. ..$ cases        : num [1:1108] 1 1 1 1 0 1 1 1 1 1 ...
##   .. .. .. .. ..$ controls     : num [1:2450] 0 0 0 0 0 0 1 1 0 0 ...
##   .. .. .. .. ..$ fun.sesp     :function (thresholds, controls, cases, direction)
##   .. .. .. .. .. ..- attr(*, "class")= chr "roc"
##   .. .. ..$ call              : language roc.default(response = as.integer(mldr_to_labels(mldr)), predicto
##   .. .. ..$ original.predictor: num [1:3558] 0 1 0 0 0 0 1 0 1 1 ...
##   .. .. ..$ original.response : int [1:3558] 0 1 0 0 0 0 1 0 1 0 ...
##   .. .. ..$ predictor         : num [1:3558] 0 1 0 0 0 0 1 0 1 1 ...
##   .. .. ..$ response          : int [1:3558] 0 1 0 0 0 0 1 0 1 0 ...
##   .. .. ..$ levels            : chr [1:2] "0" "1"
##   .. .. ..- attr(*, "class")= chr "roc"
```

```
##   ..$ call              : language roc.default(response = as.integer(mldr_to_labels(mldr)), predictor = as
##   ..$ original.predictor: num [1:3558] 0 1 0 0 0 0 1 0 1 1 ...
##   ..$ original.response : int [1:3558] 0 1 0 0 0 0 1 0 1 0 ...
##   ..$ predictor         : num [1:3558] 0 1 0 0 0 0 1 0 1 1 ...
##   ..$ response          : int [1:3558] 0 1 0 0 0 0 1 0 1 0 ...
##   ..$ levels            : chr [1:2] "0" "1"
##   ..- attr(*, "class")= chr "roc"
```

```
plot(res$ROC, main = "ROC curve for emotions") # Plot ROC curve
```

If the **pROC** [16] package is available, this list will include non-null AUC (*Area Under the ROC Curve*) measures and also a member called `ROC`. The latter holds the information needed to plot the ROC (*Receiver Operating Characteristic*) curve, as shown in the last line of the previous example. The result would be a plot similar to that in the Figure 2.

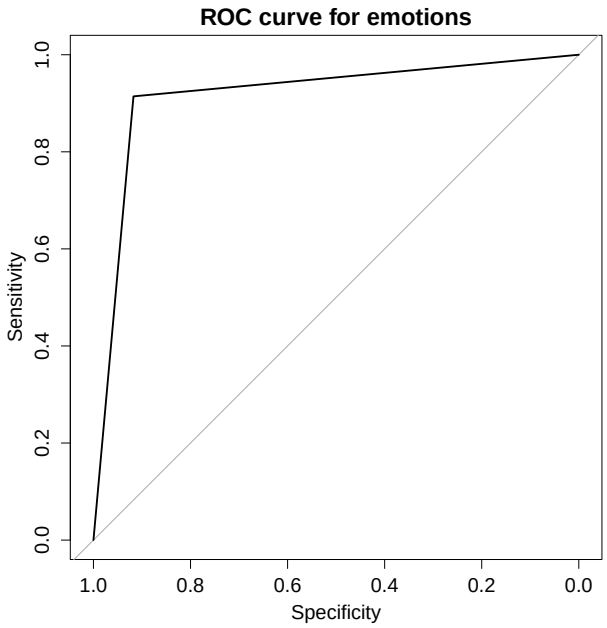


Figure 2: ROC curve plot with the data returned by `mldr_evaluate`.

4 The mldr user interface

This package provides the user with a web-based graphic user interface on top of **shiny**, allowing to interactively manipulate measurements and obtain graphics and other results. Once **mldr** is loaded, this GUI can be launched from the R console with a single command:

```
mldrGUI()
```

This will cause the user's default browser to start or open a new tab in which the GUI will be displayed, organized into a tab bar and a content pane. The tab bar allows the change of section so that different information is shown in the pane.

The GUI will initially display the Main section, as shown in Figure 3. It contains options able to select an MLD between those available, and load a new one by uploading its ARFF and XML files onto the application. On the right side, several plots are stacked. These show the amount of attributes of each type (numeric, character or label), the amount of labels per instance, the amount of instances corresponding to labels and the number of instances related to labelsets. Each plot can be saved as an image into the filesystem. Right below these graphics, some tables containing basic measures are shown. The first one lists generic measures related to the entire MLD, and is followed by measures specific to labels, such as *Card* or *Dens*. The last table shows a summary of measures for labelsets.

The Labels section contains a table enumerating each label of the MLD with its relevant details and measures: its index in the attribute list, its count and frequency, its *IRLbl* and its *SCUMBLE*. Labels in this table can be reordered using the headers, and filtered by the Search field. Furthermore, if the list is longer than the number specified in the Show field, it will be splitted into several pages. The data shown in all tables can be exported to files in several formats. On the right side, a plot shows the amount of instances that have each label. This is an interactive plot, and allows the range of labels to be manipulated.

Since relations between labels can determine the behavior of new data, studying labelsets is important in multilabel classification. Thus, the section named Labelsets provides information about them, listing each

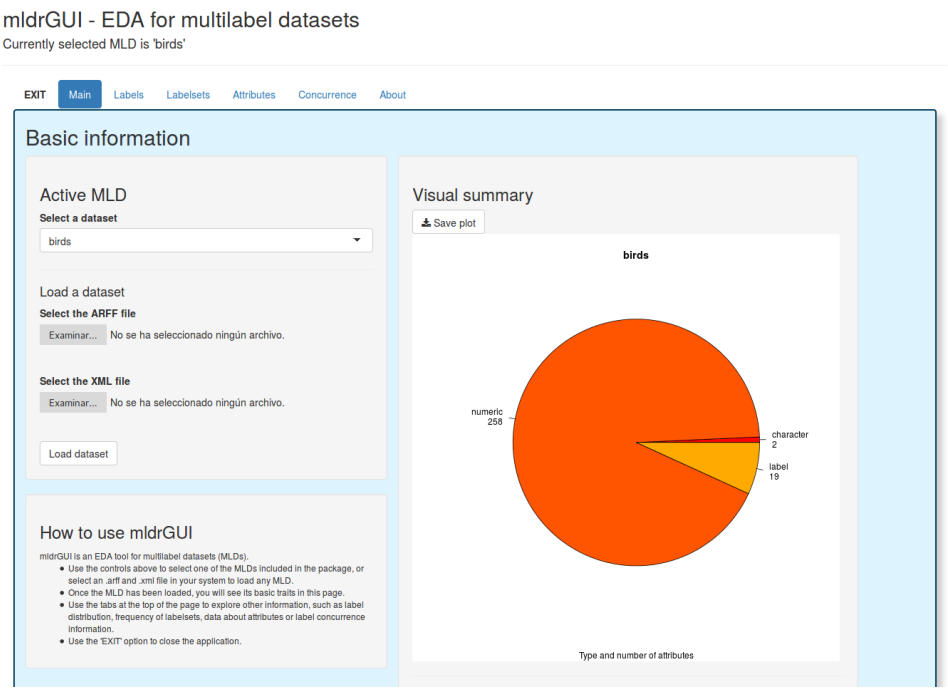


Figure 3: Main page of the shiny based user interface.

labelset along with its count. This list can be filtered and splitted into pages as well, and is accompanied by a bar plot showing the count of instances per labelset.

In order to obtain statistical measures about input attributes, the Attributes section organizes all of them into a paged table, displaying their type and some data or measures according to it. If the attribute is numeric, then there will be a table containing its minimum and maximum values, its quartiles and its mean. On the contrary, if the attribute takes values from a finite set, each possible value will be shown along with its count in the MLD.

Lastly, concurrence among labels is provenly a factor to take into account when applying preprocessing techniques to MLDs. For this reason, the Concurrence section attempts to create an easy way of visualizing concurrence among labels (see Figure 4), with a label concurrence plot displaying the selected labels in the left-side table and their coocurrences represented by bands in the circle. By default, the ten labels with higher *SCUMBLE* are selected. The user will be able to select and deselect other labels by clicking their corresponding row on the table.

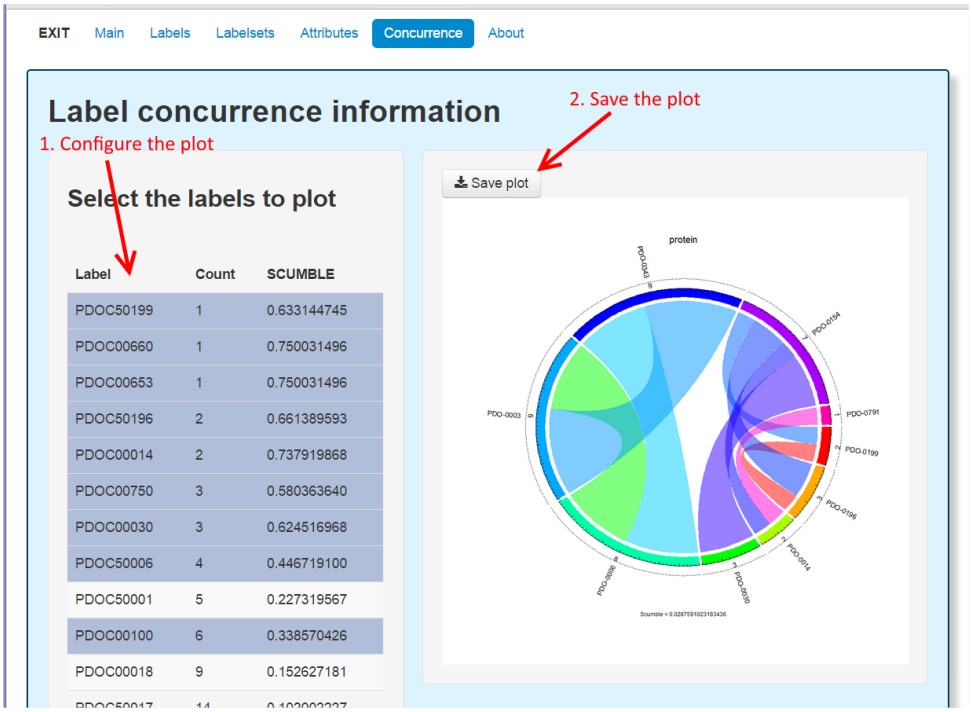


Figure 4: The plots can be customized and saved.

5 Summary

In this paper the **mldr** package, aimed to provide exploratory analysis and manipulation tools for MLDs, has been introduced. The functions supplied by this package allow both loading existent MLDs and generating new ones. Several characterization measures and specific plots can be obtained for any MLD, and its content can be extracted, filtered and joined, producing new MLDs. Any MLD can be transformed in a set of binary datasets or a multiclass dataset by mean of the transformation function of **mldr**. Finally, a web-based user interface eases the access to most of this functionality for everyone.

In its current version, **mldr** is a strong base to develop any preprocessing method for MLDs, as has been shown in a previous section. The development of the **mldr** package will continue in the near future by including the tools needed to implement and evaluate multilabel classifiers. With this foundation, we aim to encourage other developers to incorporate their own algorithms into **mldr**, as we will do in forthcoming releases.

6 Acknowledgment

This paper is partially supported by the project TIN2012-33856 of the Spanish Ministry of Science and Technology.

References

[1] M. Boutell, J. Luo, X. Shen, and C. Brown. Learning multi-label scene classification. *Pattern Recognition*, 37(9):1757–1771, 2004. ISSN 00313203. doi: 10.1016/j.patcog.2004.03.009.

[2] W. Chang. *shiny: Web Application Framework for R*, 2015. URL `http://CRAN.R-project.org/package=shiny`. R package version 0.11.

[3] F. Charte, A. Rivera, M. J. Jesus, and F. Herrera. Concurrence among imbalanced labels and its influence on multilabel resampling algorithms. In *Proc. 9th International Conference on Hybrid Artificial Intelligent Systems, Salamanca, Spain, HAIS'14*, volume 8480 of *LNCS*, 2014. ISBN 978-3-319-07616-4.

[4] F. Charte, A. J. Rivera, M. J. del Jesus, and F. Herrera. Addressing imbalance in multilabel classification: Measures and random resampling algorithms. *Neurocomputing*, 163(0):3–16, 2015. ISSN 0925-2312. doi: 10.1016/j.neucom.2014.08.091.

[5] S. Diplaris, G. Tsoumakas, P. Mitkas, and I. Vlahavas. Protein Classification with Multiple Algorithms. In *Proc. 10th Panhellenic Conference on Informatics, Volos, Greece, PCI'05*, pages 448–456, 2005. doi: 10.1007/11573036_42.

[6] P. Duygulu, K. Barnard, J. de Freitas, and D. Forsyth. Object Recognition as Machine Translation: Learning a Lexicon for a Fixed Image Vocabulary. In *Proc. 7th European Conference on Computer Vision-Part IV, Copenhagen, Denmark, ECCV'02*, pages 97–112, 2002. doi: 10.1007/3-540-47979-1_7.

[7] J. Fox. The R Commander: A basic statistics graphical user interface to R. *Journal of Statistical Software*, 14(9):1–42, 2005. URL `http://www.jstatsoft.org/v14/i09`.

[8] S. Godbole and S. Sarawagi. Discriminative Methods for Multi-Labeled Classification. In *Advances in Knowledge Discovery and Data Mining*, volume 3056, pages 22–30, 2004. doi: 10.1007/978-3-540-24775-3_5.

[9] Z. Gu, L. Gu, R. Eils, M. Schlesner, and B. Brors. circlize implements and enhances circular visualization in r. *Bioinformatics*, 30:2811–2812, 2014.

[10] K. Hornik, C. Buchta, and A. Zeileis. Open-source machine learning: R meets Weka. *Computational Statistics*, 24(2):225–232, 2009. doi: 10.1007/s00180-008-0119-7.

[11] B. Klimt and Y. Yang. The Enron Corpus: A New Dataset for Email Classification Research. In *Proc. 5th European Conference on Machine Learning, Pisa, Italy, ECML'04*, pages 217–226. 2004. doi: 10.1007/978-3-540-30115-8_22.

[12] D. T. Lang. *XML: Tools for parsing and generating XML within R and S-Plus.*, 2013. URL `http://CRAN.R-project.org/package=XML`. R package version 3.98-1.1.

[13] J. Read and P. Reutemann. MEKA: A Multi-label Extension to WEKA. URL `http://meka.sourceforge.net/`.

[14] J. Read, B. Pfahringer, and G. Holmes. Multi-label classification using ensembles of pruned sets. In *8th IEEE International Conference on Data Mining, 2008. ICDM'08.*, pages 995–1000. IEEE, 2008.

[15] J. Read, B. Pfahringer, G. Holmes, and E. Frank. Classifier chains for multi-label classification. *Machine Learning*, 85:333–359, 2011. ISSN 0885-6125. doi: 10.1007/s10994-011-5256-5.

[16] X. Robin, N. Turck, A. Hainard, N. Tiberti, F. Lisacek, J.-C. Sanchez, and M. Müller. proc: an open-source package for r and s+ to analyze and compare roc curves. *BMC Bioinformatics*, 12:77, 2011.

[17] G. Tsoumakas and I. Vlahavas. Random k-labelsets: An ensemble method for multilabel classification. In *Proc. 18th European Conference on Machine Learning, Warsaw, Poland, ECML'07*, volume 4701 of *LNCS*, pages 406–417, 2007. ISBN 978-3-540-74957-8. doi: 10.1007/978-3-540-74958-5_38.

[18] G. Tsoumakas, I. Katakis, and I. Vlahavas. Mining Multi-label Data. In O. Maimon and L. Rokach, editors, *Data Mining and Knowledge Discovery Handbook*, chapter 34, pages 667–685. Springer US, Boston, MA, 2010. ISBN 978-0-387-09822-7. doi: 10.1007/978-0-387-09823-4_34.

[19] G. Tsoumakas, E. Spyromitros-Xioufis, J. Vilcek, and I. Vlahavas. MULAN: A Java Library for Multi-Label Learning. *Journal of Machine Learning Research*, 12:2411–2414, 2011. ISSN 1532-4435.

[20] G. J. Williams. *Data Mining with Rattle and R: The art of excavating data for knowledge discovery*. Use R! Springer, 2011. URL `http://www.amazon.com/gp/product/1441998896/ref=as_li_qf_sp_asin_tl?ie=UTF8&tag=togaware-20&linkCode=as2&camp=217145&creative=399373&creativeASIN=1441998896`.

[21] M. Zhang and Z. Zhou. A review on multi-label learning algorithms. *IEEE Transanctions on Knowledge and Data Engineering*, 26(8):1819–1837, Aug 2014. ISSN 1041-4347. doi: 10.1109/TKDE.2013.39.