Basic function for peak-list manipulation and comparison

Witold E. Wolski, mail : `witek96@users.sourceforge.net`

October 28, 2005

1 Introduction

The package provides classes for storing mass spectrometric peak-lists (class `Massvector`) and collections of such lists (class `Massvectorlist`). A peak-list are mass-intensity value pairs stored in a two column `matrix` and several aditional slots *e.g.* peak-list name (`info`), target coordinates (`tcoor`), gel coordinates (`gelcoor`), and database accession string (`access`). The package provides functions for set operations (union, intersection et cetera) on `Massvector`s. The mass-spectrometric measurement has a limited accuracy. Hence, the matching of two peak-lists given a measurement error is computed. Further the package implements several dis/similarity measures on peak-lists. The peak-list matching and the pairwise dissimilarities are implemented in the C programming language. A `Massvectorlist` is able to store up to several thousand `Massvector`s. Therefore, clustering of several thousand peak-lists using this package, and the various clustering methods (*e.g.* package `stats` and `cluster`) given *e.g.* $1G$ RAM is possible.

2 Reading and Writing

First we load the library.

```
> library(msbase)
```

```
Loading required package: MASS
Loading required package: XML
```

We define some overall experiment constants like the folders with the data and the measurment error.

```
> fdat <- system.file("test/20030507a_bl_p_bla2_co1", package = "msbase")
> error <- 0.5
```

The example data was measured using a Bruker Reflex III instrument and is stored in an quite complex folder structure and a XML file (peaklist.xml). The package provides a function to read this format.

```
> mvl <- readBruker(new("Massvectorlist"), fdat)
> mvl
```

```
info     :   20030507a_bl_p_bla2_co1
project :
length   :   18
```

The package implements also a simple flat file format to store `Massvectors`. The mass, intensity value pairs are preceded by a line starting with the symbol ">" as the first character. The remainder of the line are the info-field and the target coordinates.

```
>0_A1_1SRef:1,1
832.819355960755          362.961423469165
839.227609641465          166.621956393718
842.262189000952          219.871886094500
...

> wtest <- system.file("test", package = "msbase")
> writeMV(mvl, wtest, file = "tmp.txt")
> readMV(new("Massvectorlist"), wtest, file = "tmp.txt")

info     :
project :
length   :   18
```

The loaded `Massvectorlist` contains 18 `Massvectors`. Subsets can be generated which are Massvectorlists again.

```
> mvl[1:3]

info     :   20030507a_bl_p_bla2_co1
project :
length   :   3
```

Generating a subset can be of interested if we for example want to discard peaklists with undesirable properties, *e.g.* those without peaks. This can be easily achieved using `sapply`.

```
> mvl <- mvl[which(sapply(mvl, length) != 0)]
```

Because the `Massvectorlist` extends (contains) `list` a single `Massvector` can be accessed by [[]]. The `Massvector` itself extends (contains) `Matrix`. Hence it can be subsett using [].

```
> mvl[[1]][1:10, ]

info     :   0_A10_1SRef
tcoor    :   1 10
gelcoor :   0 0
access   :
pionm    :   0
zvalue   :   2
         mass        area
1    842.375   352.6282
2    906.358   288.3520
```
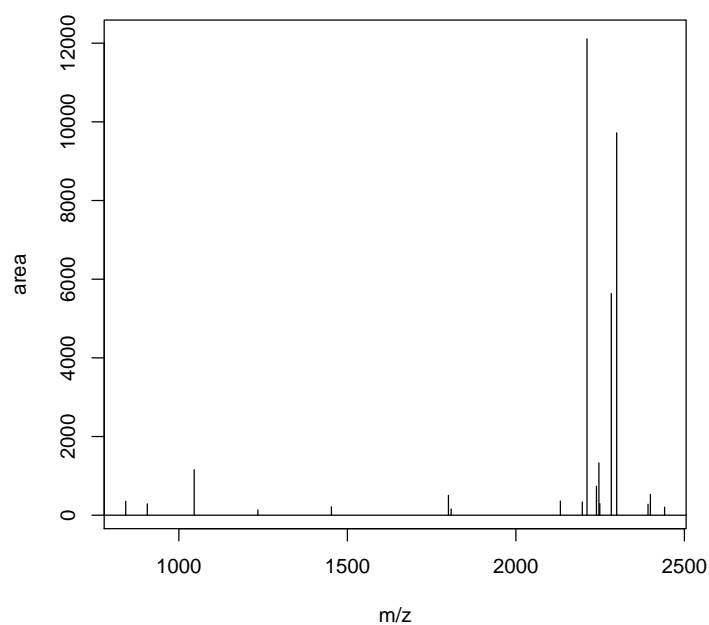
```
3   1045.453   1156.1401
4   1234.588    136.4333
5   1452.633    212.7334
6   1799.853    504.0590
7   1807.845    160.1095
8   2131.971    361.3240
9   2197.013    337.6654
10  2211.017  12109.3649
```
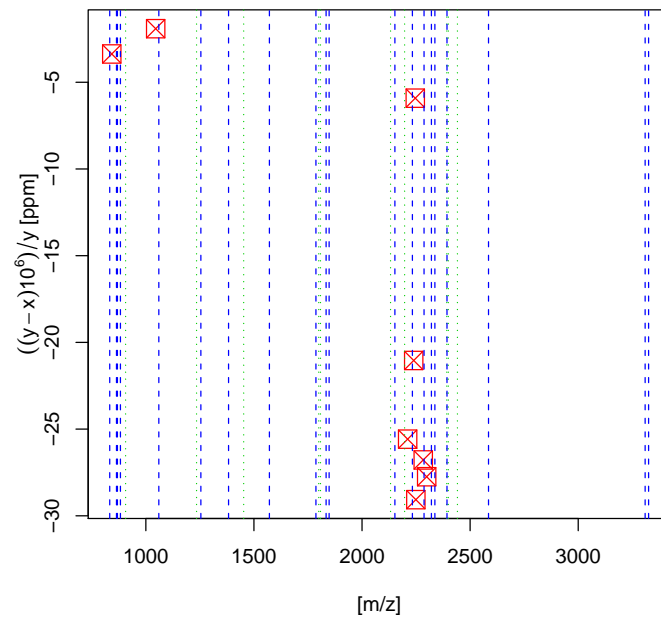
The peaks in the `Massvector` can be visualized using the `plot` and `image` function.
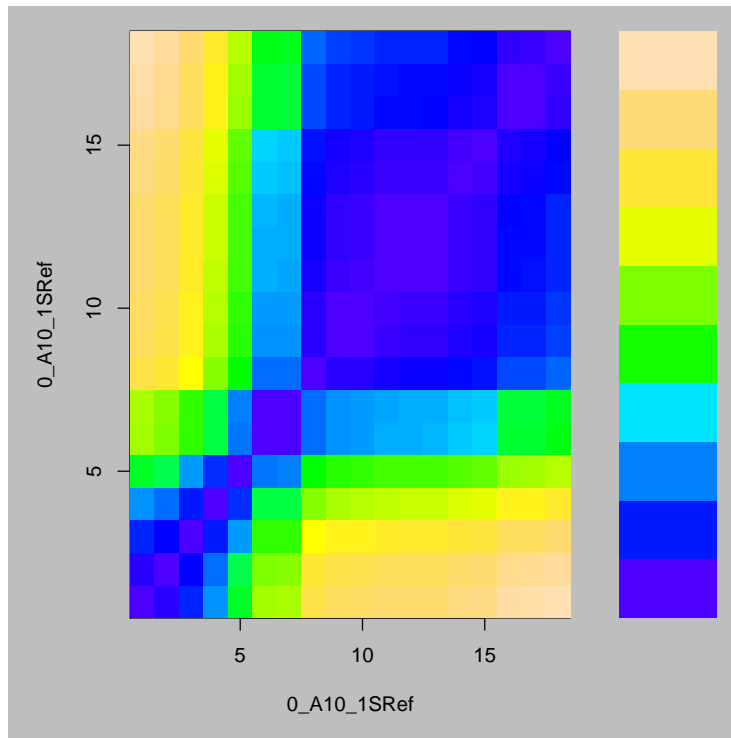
```
> plot(mvl[[1]])
```


The `plot` of a single `Massvector` is showing a stick spectrum. The `plot` of two `Massvectors` shows the residues of matching masses plotted against the average of their mass. The green and blue vertical dashed lines are drawn to show the non-matching masses.
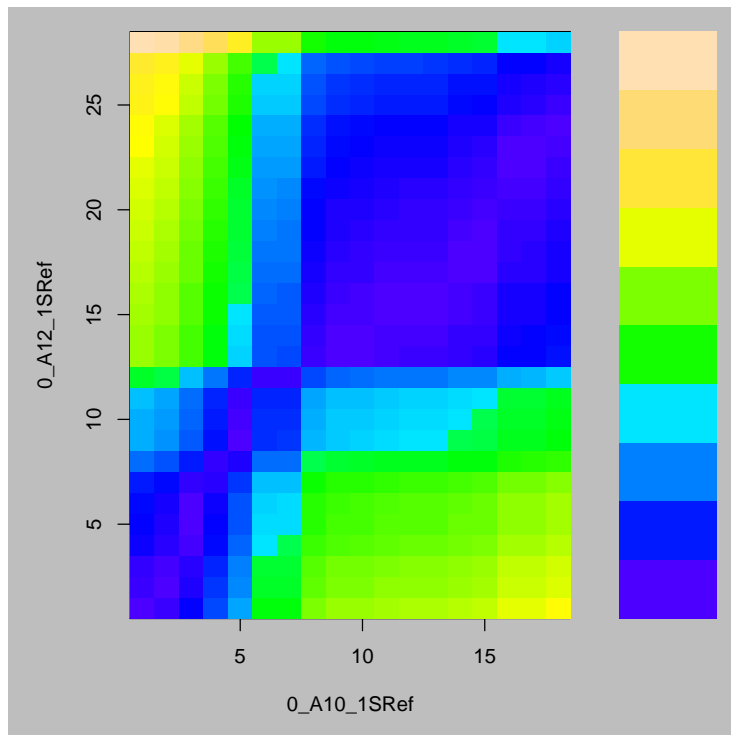
```
> plot(mvl[[1]], mvl[[2]])
```

The `image` plot color codes all mass differences between the peaks. Big blue regions are caused by peaks that are close to each other.
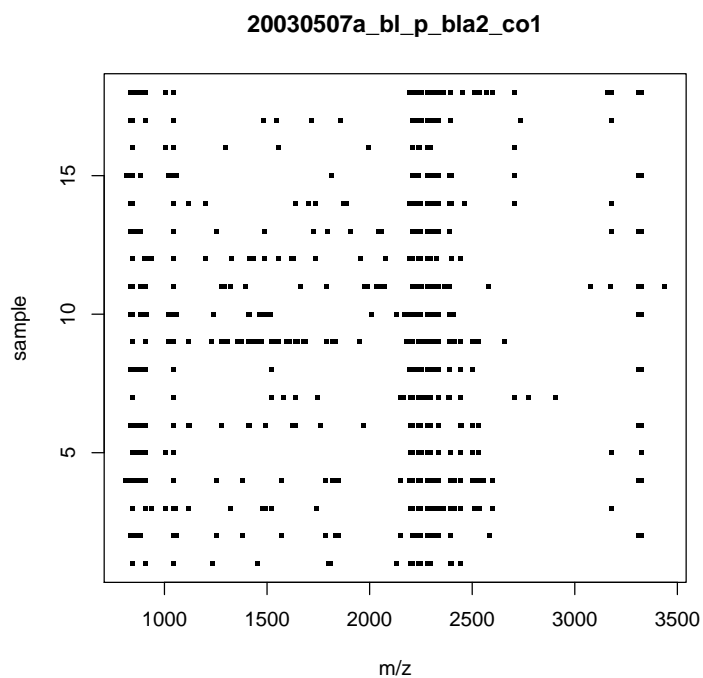
```
> image(mvl[[1]])
```

```
> image(mvl[[1]], mvl[[3]])
```
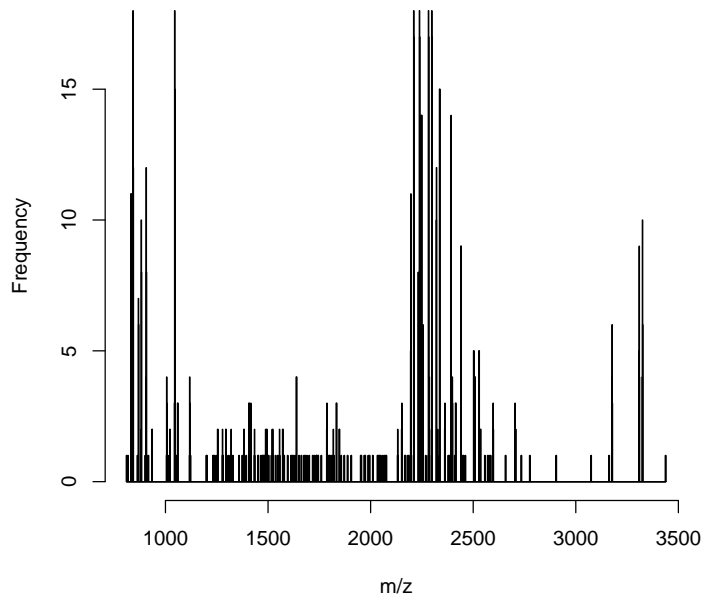

5

To take a first glance at the data in the `Massvectorlist` the methods `hist`, `plot` and `pep.plot` can be used. The `plot` function draws a strip-chart of all `Massvectors` in the `Massvectorlist`. The `hist` is showing the frequencies of all masses in the `Massvectorlist`.

```
> plot(mvl)
```
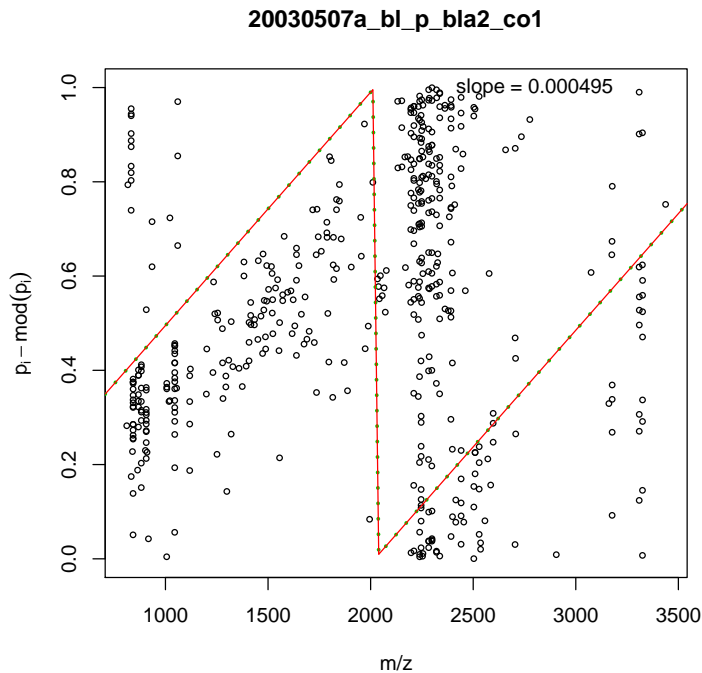
20030507a_bl_p_bla2_co1


```
> hist(mvl, accur = 2)
```

6

20030507a_bl_p_bla2_co1

The `pep.plot` plots the m mod 1 (`m%%1`) against the remainder of the peptide mass (`x-x%%1`). The linear dependence of this two values is known as peptide rule.

```
> pep.plot(mvl, cex = 0.6)
```

20030507a_bl_p_bla2_co1

A `Massvectorlist` can be transformed into R data structures:

- data.frame

```
> df <- as(mvl, "data.frame")
> names(df)

 [1] "info"         "access"       "tcoorXN"      "tcoorX"       "tcoorYN"
 [6] "tcoorY"       "gelcoorX"     "gelcoorY"     "lengthmv"     "Min.mass"
[11] "X1st.Qu.mass" "Medianmass"   "Meanmass"     "X3rd.Qu.mass" "Max.mass"
[16] "Min.int"      "X1st.Qu.int"  "Medianint"    "Meanint"      "X3rd.Qu.int"
[21] "Max.int"
```

- matrix

```
> df <- as(mvl, "matrix")
> colnames(df)

 [1] "lengthmv"     "Min.mass"     "1st Qu.mass" "Medianmass"   "Meanmass"
 [6] "3rd Qu.mass" "Max.mass"      "Min.int"      "1st Qu.int"  "Medianint"
[11] "Meanint"      "3rd Qu.int"  "Max.int"
```
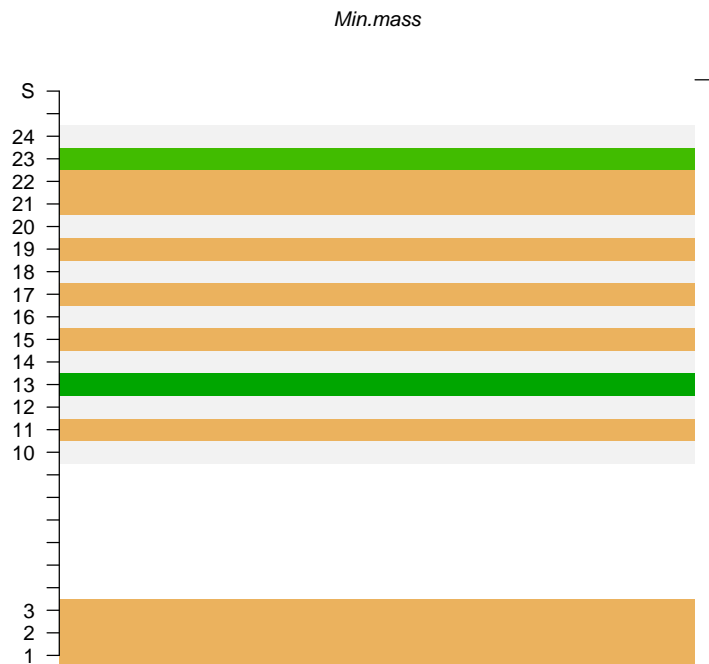
This is done by calling the `summary` function on the masses and intensities of the `Massvector`. This information can be colorcoded and visualized depending the target position. (As we mentioned each `Massvector` stores the mass spectrometric target coordinates.)

```
> mvl[[1]]@tcoor
```

```
 A 10
 1 10
```

```
> image(mvl, what = "Min.mass")
```

Min.mass

The figure above looks quite ugly because only one row from the target are provided by the sample data.

The `Massvector` can also store gel coordinates.

```
> mvl[[1]]@gelcoor
```

```
X Y
0 0
```

3   Set functions on Massvectors

There are two methods which return the indices of matching peaks. `fmatch` returns the indices of matching peaks and the residues. `fmatchall` returns the indices of matching and not matching peaks. Because this methods were first implemented to be used in calibration functions, where a peak-list is compared with a calibration list, the returned indices associated with the first argument are called `plind` (peak-list indices) while that associated with the second argument are called `calind` (calibration-list indices).

```
> pl1 <- mvl[[1]]
> pl2 <- mvl[[2]]
> fmatch(pl1, pl2, error = 400, ppm = T)
```

```
$plind
[1]  1  3 10 11 12 13 14 15 16

$calind
[1]  2  6 15 17 18 19 20 22 25

$resid
[1]  -3.368643  -1.910961 -25.579762 -21.042660  -5.917990 -29.068101 -26.778551
[8] -27.746654 380.634355

> fmatchall(pl1, pl2, error = 400, ppm = T)

$plind
[1]  1  3 10 11 12 13 14 15 16

$nplind
[1]  2  4  5  6  7  8  9 17 18

$calind
[1]  2  6 15 17 18 19 20 22 25

$ncalind
 [1]  1  3  4  5  7  8  9 10 11 12 13 14 16 21 23 24 26 27 28
```

The package provides functions to compute the union, intersection, difference, equality and membership of two `Massvectors`. A mass spectrometric measurement has an error. Therefore, the functions have additional parameters to specify the error and their names are preceded by the letter `f` (fuzzy). The error can be specified in the units of the measurement (absolute error) or as relative error in parts per million (ppm). If one peak in peak-list x matches multiple peaks in list y the optimal alignment can be computed by setting `uniq=T`.

```
> funion(pl1, pl2, error = 400, ppm = T)
> fintersect(pl1, pl2, error = 400, ppm = T, uniq = T)

info    :  0_A10_1SRef
tcoor   :  1 10
gelcoor :  0 0
access  :
pionm   :  0
zvalue  :  2
           mass freq
 [1,]   842.3735    2
 [2,]  1045.4524    2
 [3,]  2210.9885    2
 [4,]  2239.0330    2
 [5,]  2246.2961    2
 [6,]  2248.9414    2
 [7,]  2283.0686    2
 [8,]  2299.0695    2
 [9,]  2392.5028    2
```

abbreviation	name	type	range
rmi	Relative mutual information	S	$[0, 1]$
hg	Huberts Γ	S	$[-1, 1]$
fm	Fowlkes Mallows Statistiks	S	$[0, 1]$
gower	gower (jaccard) coefficient	D	$[0, 1]$

Table 1: Binary measures. type : S – similarity or D – dissimilarity.

```
> fsetdiff(pl1, pl2, error = 400, ppm = T, uniq = T)

info    :  0_A10_1SRef
tcoor   :  1 10
gelcoor :  0 0
access  :
pionm   :  0
zvalue  :  2
       mass     area
2   906.358 288.3520
4  1234.588 136.4333
5  1452.633 212.7334
6  1799.853 504.0590
7  1807.845 160.1095
8  2131.971 361.3240
9  2197.013 337.6654
17 2399.089 530.1143
18 2441.230 203.3045

> fsetequal(pl1, pl2, error = 400, ppm = T)

[1] FALSE

> fis.element(pl1, pl2, error = 400, ppm = T)

 [1]  TRUE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE
[13]  TRUE  TRUE  TRUE  TRUE FALSE FALSE
```

The `msbase` package function are equivalent to the following methods in the `base` package: `union,intersect,setdiff,setequal,is.element`.

4 Pairwise comparison of peaklists – dis/similarities

The package provides methods to compute pair-wise dis/similarities of peak-lists.

4.1 Binary measures.

The binary measures numerated in Table 1 are implemented.

The measures are in addition parametrized. We can adjust by `theta` how strongly non-matching peaks will contribute to the final score. Furthermore, the

mass agreement of matching peaks can be weighted by a triangular weighting function (`weight=T`).

$$w = \frac{a - |m_x - m_y|}{a} \tag{1}$$

m_x and m_y are the masses of matching peaks and a are the measurement accuracy.

```
> fbinary(pl1, pl2, error = 400, ppm = T, theta = 1, weight = F,
+     method = "rmi", uniq = T)

[1] -0.3488103

> fbinary(pl1, pl2, error = 400, ppm = T, theta = 0.2, weight = F,
+     method = "rmi")

[1] -0.08477856

> fbinary(pl1, pl2, error = 400, ppm = T, theta = 0.2, weight = F,
+     method = "hg")

[1] 0.05011148

> fbinary(pl1, pl2, error = 400, ppm = T, theta = 0.2, weight = T,
+     method = "gower")

[1] 0.5542329

> fbinary(pl1, pl2, error = 400, ppm = T, theta = 0.2, weight = T,
+     method = "fm")

[1] 0.7169627
```

4.2 Intensity Based dissimilarities

The package implements two groups of intensity based dis/similarities.

4.2.1 Dot product based dissimilarities

This group includes the Pearson correlation coefficient, spectral angle and spearman correlation. The only difference between the spectral angle and the Pearson correlation are the scaling of the peak intensities. The type of scaling of the intensities can be selected by setting the parameter `scale` to one of the character strings given in the first column of Table 2.

The difference between Spearman correlation and Pearson correlation are the normalization of the intensities. To compute the Spearman correlation the intensities are replaced by they ranks. This can be easily accomplished using the **R** function `rank`. Hence, no special functions are provided by the package. The following code snipped is showing how to log-transform all peak intensities in a `Massvectorlist`.

abbreviation	name
no	no scaling
tic	total ion current count
zscore	centered and divided by the variance
student	constant variance

Table 2: Scaling methods for peak intensities.

```
> mvlog <- function(x) {
+     x[, 2] <- log(x[, 2])
+     x
+ }
> as(mvl, "list") <- lapply(mvl, mvlog)
```

As for the binary measures the dot product based measures are parameterized to flexible weight the non-matching peaks and mass measurement accuracy.

```
> fcor(pl1, pl2, error = 400, ppm = T, theta = 1, weight = F, scale = "no",
+     method = "dotprod")

[1] 189694921

> fcor(pl1, pl2, error = 400, ppm = T, theta = 1, weight = F, scale = "tic",
+     method = "dotprod")

[1] 180247.9

> fcor(pl1, pl2, error = 400, ppm = T, theta = 1, weight = F, scale = "student",
+     method = "dotprod")

[1] 34.81955

> fcor(pl1, pl2, error = 400, ppm = T, theta = 1, weight = F, scale = "zscore",
+     method = "dotprod")

[1] 34.80987
```

A second method which can be accessed by this function are the sum of agreeing intensities (`method="soai"`). It puts more emphasis on the agreement of peak intensities then the correlation coefficient. All methods provided by the function `fcor` are similarity measures.

4.2.2 Distance measures

This group of measures covers the two lp-norm based measures: Euclidean and Manhattan distance, as well as the two relative distances Canberra distance, and similarity index. The measure can be chosen by setting the argument `method` to one of the characters in column one of Table 3. All other parameters which can be passed to this function are equal to those for the function `fcor`.

```
> fdist(pl1, pl2, error = 400, ppm = T, theta = 1, weight = F,
+     scale = "zscore", method = "euclidean")

[1] 1.542807
```

method	description
euclidean	Euclidean distance
manhattan	Manhattan distance
canberra	Canberra distance
simindex	similarity index

Table 3: Distance measures.

5 Searching and Clustering

Building on the pairwise `Massvector` comparison, functions to search similar spectra in `Massvectorlist`s or to cluster the `Massvector`s in `Massvectorlist` are implemented. Hence, the functions `fbinary`, `fcor`, `fdist` allow not only to compare two peak-lists but also search with one peak-list in a set of peak-lists.

```
> res <- fcor(mvl, pl1, error = 400, ppm = T, theta = 1, weight = F,
+     scale = "zscore", method = "dotprod")
```

We can find the most similar peak-list by:
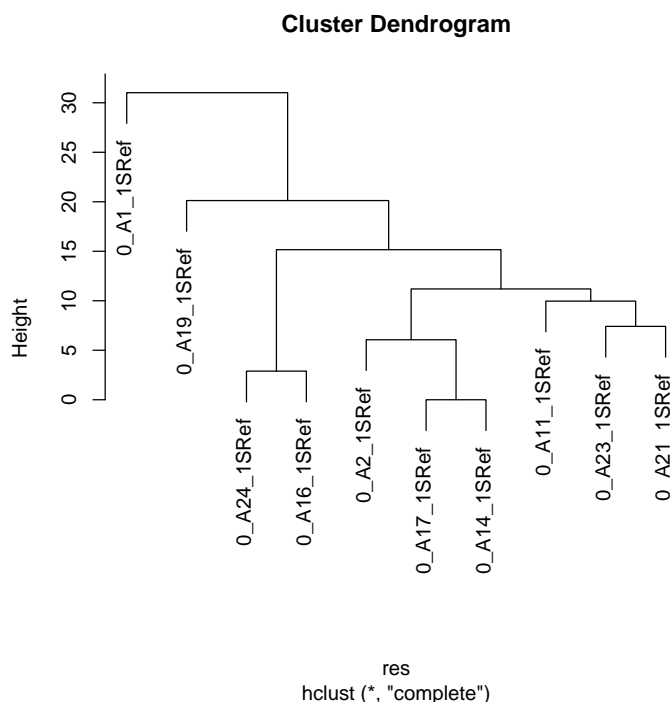
```
> which(max(res) == res)

0_A18_1SRef
          9
```

Further we can order all peak-lists according the similarity to `pl1`.

```
> mvl <- mvl[order(res)]
```

The same function interface can be used to compute a distance matrix (class `dist`) from a `Massvectorlist`. In this case the second argument must be `NULL`. An object of class `dist` can be passed to variety of clustering functions available in **R**.

```
> pp <- mvl[1:10]
> res <- fcor(pp, NULL, error = 400, ppm = T, theta = 1, weight = F,
+     scale = "zscore", method = "dotprod")
> plot(hclust(res, method = "complete"))
```

Cluster Dendrogram

res
hclust (*, "complete")

If the function `fcor` is called in this way, the similarity measures are automatically transformed into dissimilarities.

6   Conclusion

The clustering shows that peak-lists are very similar. This are mainly because of peaks caused by unspecific trypsin and matrix peaks. Furthermore the mass measurement error for this data is large. To calibrate and filter the data the package *mscalib* is developed (see : `r4proteomics.sourceforge.net`).

7   Open for contributions

This document and package is by no means complete. So if you find something missing, unclear or language errors, please send me an e-mail. I don't get offended because of this. The package is free software published under the GNU public license and is hosted on `r4proteomics.sourceforge.net` to give you the means to contribute to it. So if you think that this package is worth improvement or extending send an e-mail with your contributions. Especially functions in the file `undocumented-functions.Rd` need to be documented. Also examples to the `examples` section of the *.Rd files are wellcome. If the contributions are going to happen regularly, and you are a registered `www.sourceforge.net` user, I will grant you developer permissions so you can check in your changes into the CVS by yourself. By this your contribution will get documented.