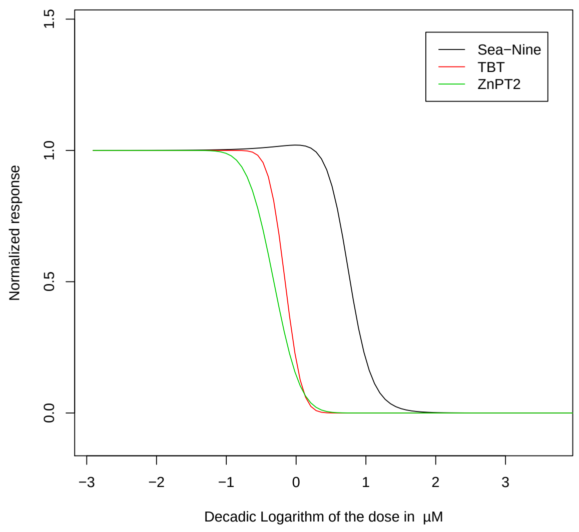

Figure 7: Dose-response plot showing the fitted dose-response curves for three antifouling biocides in the cytotoxicity assay fitted with the linlogit model.

The whole system provides the basis for analysis of the toxicity data, e.g. by (Quantitative) Structure-Activity Relationships (SAR/QSAR), which may provide a deeper chemical understanding of the interaction of the chemicals with biological organisms.

Bibliography

D. M. Bates and D. G. Watts. *Nonlinear Regression Analysis and its Applications*. Wiley Series in Probability and Mathematical Statistics. Wiley, New York, 1988. 7

P. Brain and R. Cousens. An equation to describe dose responses where there is stimulation of growth at low doses. *Weed Research*, 29:93–96, 1989. 10

J. Ranke, K. Mölter, F. Stock, U. Bottin-Weber, J. Poczobutt, J. Hoffmann, B. Ondruschka, J. Filser, and B. Jastorff. Biological effects of imidazolium ionic liquids with varying chain lengths in acute Vibrio fischeri and WST-1 cell viability assays. *Ecotoxicology and Environmental Safety*, 28(3):396–404, 2004. 9

P. H. van Ewijk and J. A. Hoekstra. Calculation of the EC50 and its confidence interval when subtoxic stimulus is present. *Ecotoxicology and Environmental Safety*, 25:25–32, 1993. 10

W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Statistics and Computing. Springer, New York, 2002. 7

Johannes Ranke
Department of Bioorganic Chemistry
UFT Center for Environmental Research and Technology
University of Bremen jranke@uni-bremen.de

The pls package

by Bjørn-Helge Mevik

Introduction

The **pls** package implements *Partial Least Squares Regression* (PLSR) and *Principal Component Regression* (PCR). It is written by Ron Wehrens and Bjørn-Helge Mevik.

PCR is probably well-known to most statisticians. It consists of a linear regression of one or more responses **Y** onto a number of principal component scores from a predictor matrix **X** (Næs and Martens, 1988).

PLSR is also a linear regression onto a number of components from **X**, but whereas principal component analysis maximizes the variance of the scores, PLS maximizes the covariance between the scores and the response. The idea is that this should give components that are more relevant for the response. Typically, PLSR achieves the same (or smaller) prediction error as PCR, with fewer components. A good introduction to PLSR and PCR can be found in Martens and Næs (1989). A review of PLSR is given in Wold et al. (2001) (in fact, all of that issue of Chemolab is dedicated to PLSR). Frank and Friedman (1993) provides a more technical treatment, from a statistical viewpoint.

PLSR and PCR are commonly used in situations where there are collinearities or near-collinearities in **X**, for instance when there are more variables than observations. This is a very common situation in fields like chemometrics, where various types of spectroscopic data are often used to predict other measurements.

There are other regression methods that can be applied to such data, for instance ridge regression. Studies have indicated that in terms of prediction error, ridge regression can perform slightly better than PLSR. However, one of the major advantages of PLSR and PCR is interpretation. In addition to a prediction equation, one gets score and loading vectors for each component. These can be plotted and interpreted by the field expert. There is a strong focus on graphical assessment and interpretation in fields like chemometrics.

Typical usage

To illustrate the usage of the package, we will use a data set published in Kalivas (1997). It consists of octane number and *near infrared* (NIR) spectra of 60 gasoline samples. Each NIR spectrum consists of 401 diffuse reflectance measurements from 900 to 1700 nm. The spectra are shown in Figure 1.

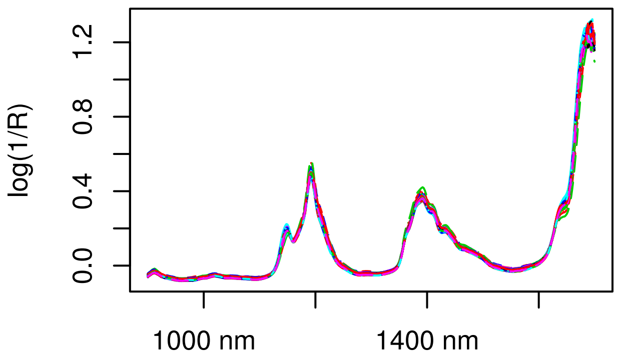

Figure 1: NIR spectra

The user interface is inspired by `lm`, and has a formula interface and methods for `plot`, `predict`, `fitted`, etc. The main modelling functions are `plsr` and `pcr`, which fit PLSR and PCR models, respectively. They are both simple wrappers for `mvr`, and return a fitted model as an object of class `"mvr"`. The object usually contains fit results for models with 0, 1, . . ., `ncomp` components, where `ncomp` is specified in the call. Because the score vectors are mean centered and orthogonal, one can inspect submodels by selecting subsets of the components, without having to refit the model. Many extraction and plot functions support this through the argument `comps` (or, for a few functions, `ncomp`).

It is customary to use *cross-validation* (CV) (Stone, 1974) or test set validation to determine how many components to use in the regression. The modelling functions can perform cross-validation, and the results are stored in the model object.

We will illustrate the use of **pls** by doing a PLSR analysis of the gasoline data. A typical way of calling `plsr` is:

```
> gas1 <- plsr(oct ~ NIR, ncomp = 10,
+     data = gasoline, validation = "LOO")
```

This fits a model with 10 components, and includes *leave-one-out* (LOO) cross-validated predictions (Lachenbruch and Mickey, 1968). One can extract different elements of the fitted model with functions like `coef`, `scores`, `loadings`, etc.

The `print` method for `"mvr"` objects shows the type of model and fit algorithm, while the `summary` method gives more details about the fit and validation results (if any):

```
> summary(gas1)

Data:   X dimension: 60 401
        Y dimension: 60 1
Fit method: kernelpls
Number of components considered: 10

VALIDATION: RMSEP
Cross-validated using 60 leave-one-out segs.
       (Intercept)  1 comps  2 comps
CV           1.543    1.328   0.3813
adjCV        1.543    1.328   0.3793
       3 comps  4 comps  5 comps  6 comps
CV      0.2579   0.2412   0.2412   0.2294
adjCV   0.2577   0.2410   0.2405   0.2288
       7 comps  8 comps  9 comps  10 comps
CV      0.2191   0.2280   0.2422    0.2441
adjCV   0.2183   0.2273   0.2411    0.2433

TRAINING: % variance explained
     1 comps  2 comps  3 comps  4 comps
X      70.97    78.56    86.15     95.4
oct    31.90    94.66    97.71     98.0
     5 comps  6 comps  7 comps  8 comps
X      96.12    96.97    97.32     98.1
oct    98.68    98.93    99.06     99.1
     9 comps  10 comps
X      98.32     98.71
oct    99.20     99.24
```

The validation results here are *Root Mean Squared Error of Prediction* (RMSEP). There are two cross-validation estimates: 'CV' is the ordinary CV estimate, and 'adjCV' is a bias-corrected CV estimate (Mevik and Cederkvist, 2004) (For a LOO CV, there is virtually no difference).

As an alternative, one can plot the RMSEPs in order to decide how many components to use. The function `RMSEP` returns an object of class `"mvrVal"`, with its own `print` and `plot` methods. `RMSEP` can also calculate a test set estimate, if a test set is provided with the `newdata` argument. (If one prefers MSE, one can use the function `MSEP`.)

```
> plot(RMSEP(gas1), legendpos = "topright")
```

This plots the estimated RMSEPs as functions of the number of components (Figure 2). The `legendpos` argument adds a legend at the indicated position. Three components seem to be enough. This gives an RMSEP of 0.258. As mentioned in the introduction, the main practical difference between PCR and PLSR is that PCR often needs more components than PLSR to achieve the same prediction error. On this data set, PCR would need four components to achieve the same RMSEP.

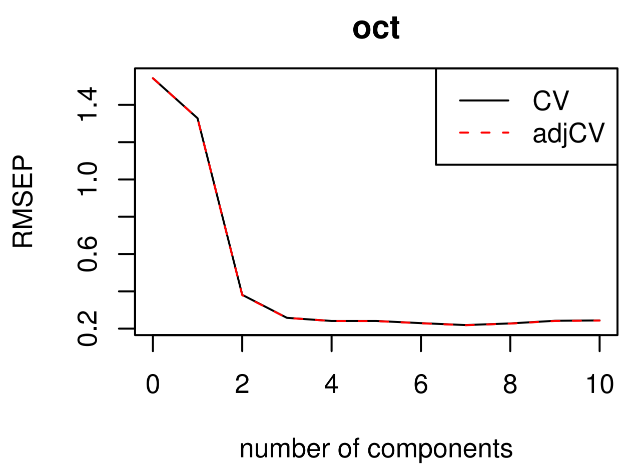

Figure 2: Cross-validated RMSEP curves

Once the number of components has been chosen, one can inspect different aspects of the fit by plotting predictions, scores, loadings, etc. The default plot is a prediction plot:

```
> plot(gas1, ncomp = 3, asp = 1, line = TRUE)
```

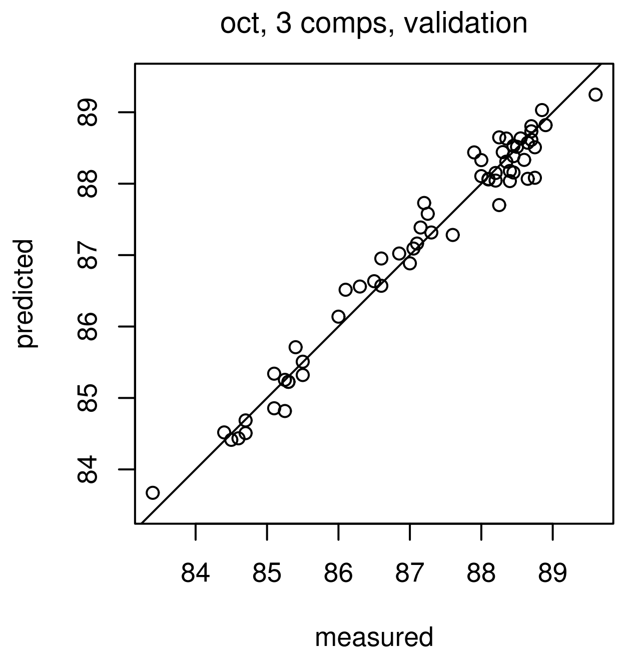

Figure 3: Cross-validated predictions

This shows the cross-validated predictions with three components versus measured values (Figure 3). We have chosen an aspect ratio of 1, and to draw a target line. One can plot fitted values instead of cross-validated predictions, by using the argument `which = "train"`.

Other plots can be selected with the argument `plottype`:

```
> plot(gas1, plottype = "scores",
+      comps = 1:3)
```

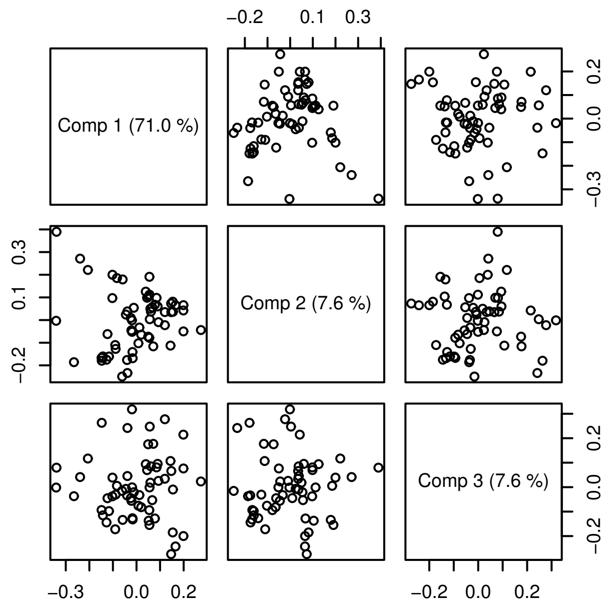

Figure 4: Score plot

This gives a pairwise plot of the score values (Figure 4). Score plots are much used to look for patterns, groups or outliers in the data. (For instance, plotting the two first components for a model built on the `NIR` dataset included in **pls**, clearly indicates the experimental design of the data.) The numbers in parentheses after the component labels are the relative amount of **X**-variance explained by each component. The explained variances can be extracted explicitly with `explvar(gas1)`.

Another common plot is the loading plot (Figure 5), which is much used for component interpretation, by looking for known spectral peaks or profiles:

```
> plot(gas1, "loadings", comps = 1:3,
+      legendpos = "topleft")
> abline(h = 0)
```

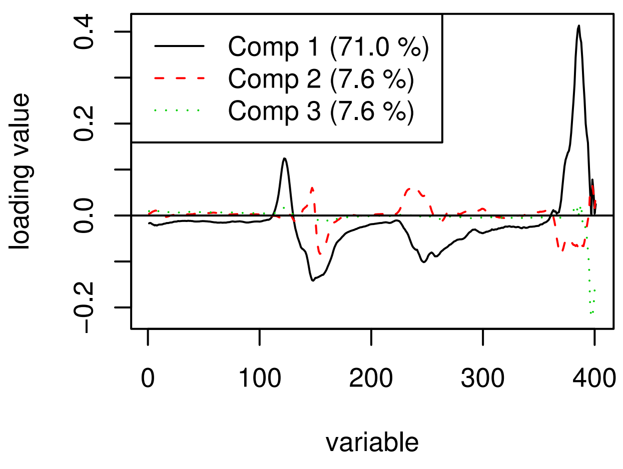

Figure 5: Loading plot

The package also implements biplots and correlation loadings plots. All the plots can also be accessed directly with functions `predplot`, `scoreplot`, etc.

Multi-response models

Multi-response models are handled automatically by the fit, extraction and plot functions in **pls**. The package contains a small sensory data set with 5 quality parameters `Quality` and 6 sensory attributes `Panel` for 16 olive oil samples. To see whether the sensory attributes can be explained by the quality measurements, one can do a regression of `Panel` on `Quality`:

```
> data(sensory)
> olive1 <- plsr(Panel ~ Quality,
+     data = sensory, val = "LOO")
```

The `plot` method for `"mvrVal"` objects gives one validation curve for each response (Figure 6):

```
> plot(RMSEP(olive1), nCols = 2)
```

The `nCols = 2` argument tells the underlying `plot.mvrVal` function to use two columns instead of three for the panels. The syrup attribute is best explained by a single component, but overall, two components seem optimal. The reduction in RMSEP is quite moderate for all responses, which is not unusual for sensory data. This can also be seen in a prediction plot (Figure 7):

```
> plot(olive1, ncomp = 2, asp = 1,
+     line = TRUE, nCols = 2)
```

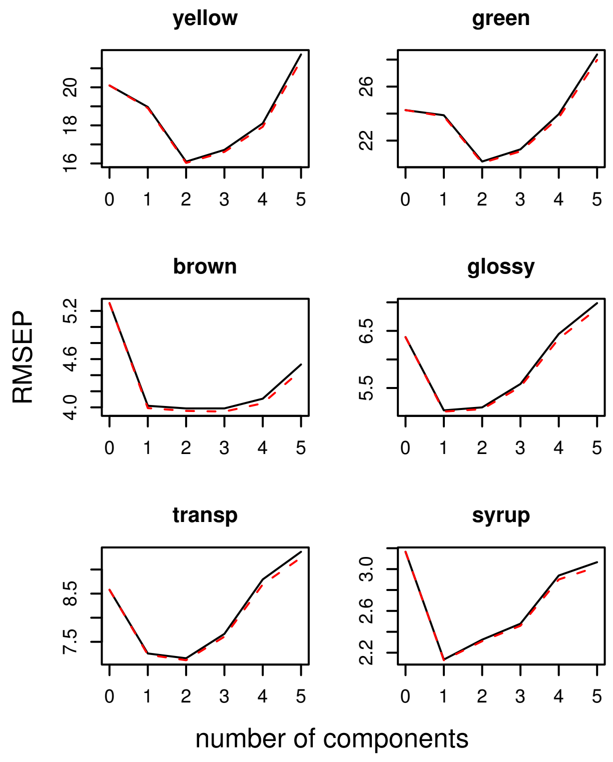

Figure 6: Cross-validated RMSEP curves

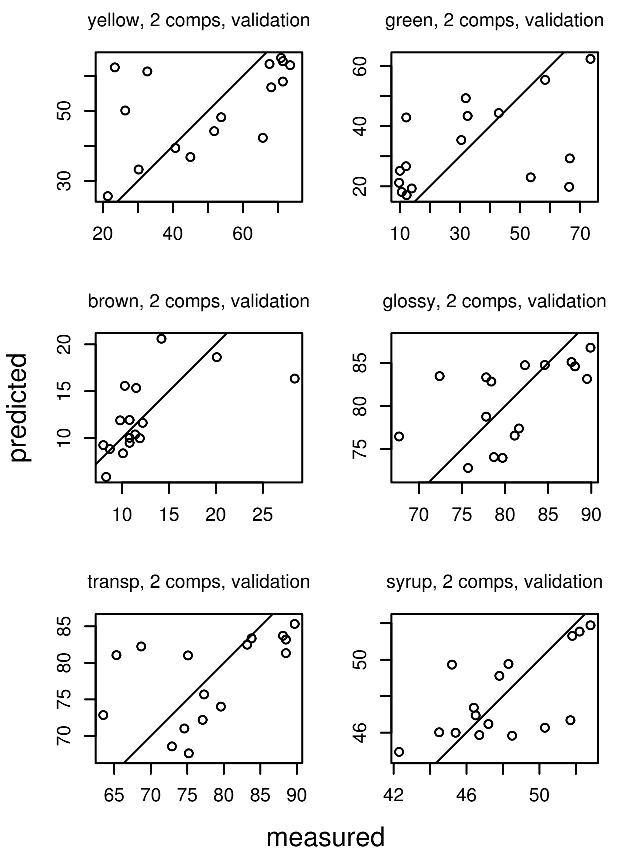

Figure 7: Cross-validated predictions

A correlation loadings plot can tell us which predictors are adequately described by the model, and which predictors correspond to the different components:

```
> corrplot(olive1, comps = 1:2,
+     labels = "names")
```

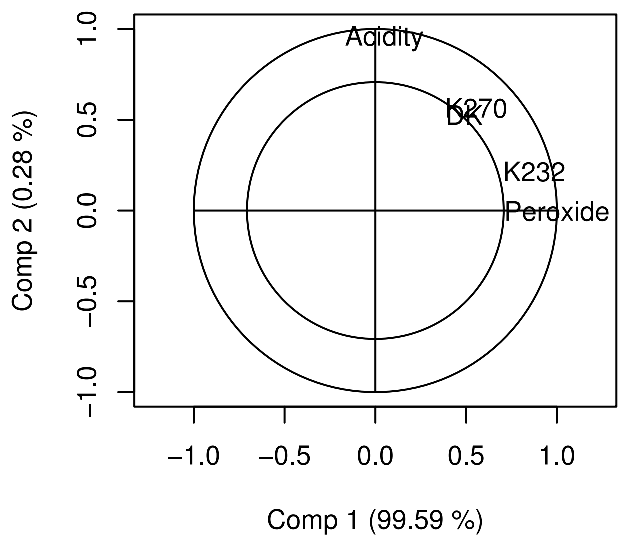

Figure 8: Correlation loadings plot

Figure 8 indicates that all predictors are quite well described, and that Peroxide and K232 are most correlated with component 1, while Acidity corresponds to the second component. The `labels = "names"` argument labels the points with the variable names. They can also be labelled with numbers. The default is the standard plot symbol. The percentages give the amount of **X**-variance explained by each component. We see that the first component explains almost all variance of **X**. However, the RMSEPs in Figure 6 show that two components are needed to explain all responses adequately.

Flexible cross-validation

The cross-validation in **pls** can handle quite complex situations, with respect to both the segments and the treatment of the data.

Often, the observations in a data set are grouped in some way or another, for instance when there are replicates. In such cases, each cross-validation segment should consist of entire groups. As an example, assume that the gasoline data actually consists of three replicate measurements performed on 20 samples, and that the replicates are stored in consecutive rows in the data table. This can be handled with

```
> gas2 <- update(gas1, validation = "CV",
+     segment.type = "consecutive",
+     length.seg = 3)
```

which gives 20 consecutive segments of size three. One can also choose interleaved or randomly selected (the default) segments. If one specifies `length.seg`, the number of segments are calculated, and *vice versa* if `segments` is specified. Care is taken so that the segment sizes differ at most by 1.

For full flexibility, e.g. for unequal segment sizes, the segments can be specified explicitly, as a list of index vectors. See `?mvrCv` for details.

In spectroscopic applications, the spectra often have to be preprocessed in order to eliminate artefacts such as light scattering effects. This is often done by taking the first or second derivative of the spectra, or using *Multiplicative Scatter Correction* (MSC). Currently, **pls** has a function `msc` which implements MSC. It can be used directly in model formulas: `gas2 <- plsr(oct ~ msc(NIR), ...)`. This is implemented such that the same preprocessing is applied to new data in `predict(gas2, newdata = new.data)`.

Formulas like this pose a problem for the built-in cross-validation: For efficiency reasons, the formula is evaluated only once in `mvr`, on the complete data set, even when the built-in CV is used. However, the scatter correction ought to be re-calculated for each CV segment, because it depends on the whole data set. This can be done by using the more general (but slower) function `crossval` for cross-validation. It takes an `"mvr"` object and returns a cross-validated object:

```
> gas2 <- plsr(oct ~ msc(NIR), ncomp = 6,
+     data = gasoline)
> gas2 <- crossval(gas2, length.seg = 1)
> RMSEP(gas2)

       (Intercept)  1 comps  2 comps
CV           1.543    1.296   0.3975
adjCV        1.543    1.296   0.3971
       3 comps  4 comps  5 comps  6 comps
CV      0.2516   0.2400   0.2212   0.2266
adjCV   0.2514   0.2389   0.2208   0.2263
```

As can be seen, there seems to be little effect of the MSC on these data. In fact, Figure 1 indicates that there is little scatter in the spectra.

Internals and Extensions

There are quite a few algorithms for calculating PLS scores and loadings. Most of them are equivalent for single-response models, and give more or less the same results for multi-response models. The **pls** package currently implements three of the more well-known algorithms: kernel PLS, SimPLS and the orthogonal scores algorithm.

Under the hood, `mvr` first handles the formula and data, and then sends the data to an underlying fit function, as determined by the `method` argument.

These fit functions can be called directly, which can be useful when speed is important, or when using PLSR or PCR as building blocks in other algorithms. As a simple example, a “quick ’n dirty” cross-validation can be made like this:

```
> X <- gasoline$NIR
> y <- gasoline$oct
> n <- nrow(X)
> A <- 10
> cvpreds <- matrix(nrow = n, ncol = A)
> for (i in 1:n) {
+     fit <- kernelpls.fit(X[-i,], y[-i],
+         ncomp = A, stripped = TRUE)
+     for (j in 1:A)
+         cvpreds[i,j] <- (X[i,] -
+             fit$Xmeans) %*%
+             fit$coefficients[,,j]
+     cvpreds[i,] <- cvpreds[i,] + fit$Ymeans
+ }
> sqrt(colMeans((y - cvpreds)^2))

 [1] 1.3281674 0.3813088 0.2578943
 [4] 0.2411522 0.2411555 0.2294477
 [7] 0.2191377 0.2279735 0.2421662
[10] 0.2440551
```

This is of course identical to the LOO CV results in `summary(gas1)`.

Another example can be found in the package **lspls**, which is available on CRAN. **lspls** uses ordinary least squares regression and PLSR to fit a response to a sequence of matrices (Jørgensen et al., 2005).

Bibliography

I. E. Frank and J. H. Friedman. A statistical view of some chemometrics regression tools. *Technometrics*, 35(2):109–148, 1993. 12

K. Jørgensen, B.-H. Mevik, and T. Næs. Combining designed experiments with several blocks of spectroscopic data. Submitted, 2005. 17

J. H. Kalivas. Two data sets of near infrared spectra. *Chemometrics and Intelligent Laboratory Systems*, 37: 255–259, 1997. 13

P. A. Lachenbruch and M. R. Mickey. Estimation of error rates in discriminant analysis. *Technometrics*, 10(1):1–11, 1968. 13

H. Martens and T. Næs. *Multivariate Calibration*. Wiley, Chichester, 1989. 12

B.-H. Mevik and H. R. Cederkvist. Mean squared error of prediction (MSEP) estimates for principal component regression (PCR) and partial least squares regression (PLSR). *Journal of Chemometrics*, 18(9):422–429, 2004. 13

T. Næs and H. Martens. Principal component regression in NIR analysis: Viewpoints, background details and selection of components. *Journal of Chemometrics*, 2:155–167, 1988. 12

M. Stone. Cross-validatory choice and assesment of statistical predictions. *Journal of the Royal Statistical Society, Series B—Methodological*, 36:111–147, 1974. 13

S. Wold, M. Sjöström, and L. Eriksson. PLS-regression: a basic tool of chemometrics. *Chemometrics and Intelligent Laboratory Systems*, 58(2):109–130, 2001. 12

Bjørn-Helge Mevik
IKBM, Norwegian University of Life Sciences,
Ås, Norway.
bjorn-helge.mevik@umb.no

Some Applications of Model-Based Clustering in Chemistry

by Chris Fraley and Adrian E. Raftery

Interest in clustering has experienced a recent surge due to the emergence of new areas of application. Prominent among these is the analysis of images resulting from new technologies involving chemical processes, such as microarray or proteomics data, and contrast-enhanced medical imaging. Clustering is applied to the image data to produce segmentations that are appropriately interpretable. Other applications include minefield detection (Dasgupta and Raftery 1998; Stanford and Raftery 2000), finding flaws in textiles (Campbell et al. 1997; 1999), grouping coexpressed genes (Yeung et al. 2001), *in vivo* MRI of patients with brain tumors (Wehrens et al. 2002), and statistical process control (Thissen et al. 2005).

The use of clustering methods based on probability models rather than heuristic procedures is becoming increasingly common due to recent advances in methods and software for model-based clustering, and the fact that the results are more easily intepretable. Finite mixture models (McLachlan and Peel, 2000), in which each component probability corresponds to a cluster, provide a principled statistical approach to clustering. Models that differ in the number of components and/or component distributions can be compared using statistical criteria. The clustering process estimates a model for the data that allows for overlapping clusters, as well as a probabilistic clustering that quantifies the uncertainty of observations belonging to components of the mixture.

The R package **mclust** (Fraley and Raftery 1999, 2003) implements clustering based on normal mixture models. The main clustering functionality is provided by the function `EMclust`, together with its associated `summary` and `plot` methods. Users can specify various parameterizations of the variance or covariance of the normal mixture model, including spherical and diagonal models in the multivariate case, along with the desired numbers of mixture components to consider. The mixture parameters are estimated via the EM algorithm (Dempster et al. 1977; McLachlan and Krishnan 1997), initialized by model-based hierarchical clustering (Banfield and Raftery 1993; Fraley 1998). The best model is selected according to the Bayesian Information Criterion or BIC (Schwarz 1978), a criterion that adds a penalty to the loglikelihood that increases with the number of parameters in the model.

In this article, we discuss an application of model-