HHS Public Access

Author manuscript

J Stat Softw. Author manuscript; available in PMC 2018 November 16.

Published in final edited form as:

J Stat Softw. 2018 April ; 84: . doi:10.18637/jss.v084.i01.

clustvarsel: A Package Implementing Variable Selection for Gaussian Model-Based Clustering in R

Luca Scrucca and

Department of Economics, Università degli Studi di Perugia, Via A. Pascoli, 20, 06123 Perugia, Italy, URL: http://www.stat.unipg.it/luca

Adrian E. Raftery

Department of Statistics, University of Washington, Box 354320, Seattle, WA 98195-4320, United States of America, URL: http://www.stat.washington.edu/raftery/

Abstract

Finite mixture modeling provides a framework for cluster analysis based on parsimonious Gaussian mixture models. Variable or feature selection is of particular importance in situations where only a subset of the available variables provide clustering information. This enables the selection of a more parsimonious model, yielding more efficient estimates, a clearer interpretation and, often, improved clustering partitions. This paper describes the R package **clustvarsel** which performs subset selection for model-based clustering. An improved version of the Raftery and Dean (2006) methodology is implemented in the new release of the package to find the (locally) optimal subset of variables with group/cluster information in a dataset. Search over the solution space is performed using either a step-wise greedy search or a headlong algorithm. Adjustments for speeding up these algorithms are discussed, as well as a parallel implementation of the stepwise search. Usage of the package is presented through the discussion of several data examples.

Keywords

BIC; model-based clustering; R; subset selection

1. Introduction

Cluster analysis is the search for a priori unknown group structure in data. Model-based clustering is increasingly becoming one of the most popular cluster analysis methods. Model-based clustering is based on finite mixture models (McLachlan and Peel 2000), with each component density usually representing a cluster. For continuous data, Gaussian components are usually used to model clusters. Model-based clustering as implemented in the R package **mclust** (Fraley, Raftery, Murphy, and Scrucca 2012; Scrucca, Fop, Murphy, and Raftery 2016) allows for automatic selection of the number of components, and selection of parsimonious covariance structures.

In cluster analysis, as in classification or other supervised learning tasks, the inclusion of noise variables, i.e., features without useful group information, can severely degrade the final results. In fact, the presence of noise variables can negatively impact both the

estimation of the number of clusters in the data and the recovery of those groups. The new release of R package **clustvarsel** (version ≥ 2.0; Dean, Raftery, and Scrucca 2017) implements a wrapper method for automatic variable selection in model-based clustering (as implemented in the **mclust** package). Thus, the addition of the **clustvarsel** package allows for automatic variable selection to be included in the estimation process.

Raftery and Dean (2006) introduced a stepwise variable selection methodology tailored to model-based clustering. Variables designated as noise variables in this process were not required to be independent of the clustering variables. However, noise variables could be conditionally independent of the clustering, but still linearly dependent on the clustering variables. This linear dependency was modeled using linear regression. An earlier version of **clustvarsel** (version 1) implemented this methodology. Dean (2006) is a vignette describing use of this earlier version.

Maugis, Celeux, and Martin-Magniette (2009a, b) extended the framework of Raftery and Dean (2006) by allowing the noise variables to depend on a (possibly null) subset of the clustering variables via stepwise variable selection in the linear regression. This allows for a more parsimonious modeling of the relationship between the noise variables and the clustering variables. For more details on the variable selection framework see Section 2.

Software packages related to subset selection in clustering are **SelvarClust** (Dia, Martin-Magniette, and Maugis 2009a) and **SelvarClustIndep** (Dia, Martin-Magniette, and Maugis 2009b), which implement in C++ the above mentioned approaches. The R package **SelvarMix** (Sedki, Celeux, and Maugis-Rabusseau 2017) provides a method based on the Maugis *et al.* (2009b) approach preceded by a step in which the variables are ranked using a lasso-like procedure. The R package **vscc** (Andrews and McNicholas 2013) implements the methodology proposed by Andrews and McNicholas (2014) which aims at finding the variables that simultaneously minimize the within-group variance and maximize the between-group variance. Finally, sparse hierarchical clustering and sparse k-means clustering are included in the R package **sparcl** (Witten and Tibshirani 2013) according to the proposal of Witten and Tibshirani (2010).

The paper is organized as follows: Section 3 introduces the main function in the **clustvarsel** package, and discusses the options for the available arguments. In Section 4, several examples are presented by applying the methodology to both synthetic and real world datasets. Algorithmic speedups are discussed in Section 5, including a description of a parallel implementation of the stepwise greedy search. The paper concludes with some discussion and final remarks in Section 6.

2. Methodology

Model-based clustering assumes that the observed data are generated from a mixture of G components, each representing the probability distribution for a different group or cluster (McLachlan and Peel 2000; Fraley and Raftery 2002). For continuous data, the density of each mixture component is often described by the multivariate Gaussian distribution. Thus, the general form of a Gaussian finite mixture model is

$$f(\boldsymbol{x}) = \sum_{g=1}^{G} \pi_g \phi(\boldsymbol{x} | \boldsymbol{\mu}_g, \boldsymbol{\Sigma}_g),$$

where $\boldsymbol{\pi}_g$ represents the mixing probabilities, so that $\boldsymbol{\pi}_g > 0$ and $\sum_{g=1}^{G} \pi_g = 1$, $\phi(\cdot)$ is the multivariate Gaussian density with parameters $(\boldsymbol{\mu}_g, \boldsymbol{\Sigma}_g)$ ($g = 1, \ldots, G$). Clusters are ellipsoidal, centered at the mean vector $\boldsymbol{\mu}_g$, with other geometric features, such as volume, shape and orientation, determined by $\boldsymbol{\sigma}_g$. Parsimonious parameterization of covariance matrices is available through the eigenvalue decomposition $\boldsymbol{\Sigma}_g = \lambda_g \boldsymbol{D}_g \boldsymbol{A}_g \boldsymbol{D}_g^\top$, where λ_g is a scalar controlling the volume of the ellipsoid, $\boldsymbol{A}_g$ is a diagonal matrix specifying the shape of the density contours, and $\boldsymbol{D}_g$ is an orthogonal matrix which determines the orientation of the corresponding ellipsoid (Banfield and Raftery 1993; Celeux and Govaert 1995). Fraley *et al.* (2012, Table 1) report some parameterization of within-group covariance matrices available in the R package **mclust**, and the corresponding geometric characteristics.

Raftery and Dean (2006) discussed the problem of variable selection for model-based clustering by recasting the problem as a model selection procedure. Their proposal is based on the use of the Bayesian information criterion (BIC) to approximate Bayes factors to compare mixture models fitted on nested subsets of variables. A generalization of their approach was later discussed by Maugis *et al.* (2009a, b).

Let us suppose that the set of available variables, χ, is partitioned into three disjoint parts: the set of previously selected variables, χ_{clust}, the variable under consideration for inclusion or exclusion from the active set, X_i, and the set of the remaining variables, $\chi_{\text{other}} \equiv \chi \setminus \{\chi_{\text{clust}} \cup X_i\}$.

Raftery and Dean (2006) showed that the inclusion (or exclusion) of variables can be assessed using the following BIC difference:

$$\text{BIC}_{\text{diff}} = \text{BIC}_{\text{clust}}(\chi_{\text{clust}}, X_i) - \text{BIC}_{\text{not clust}}(\chi_{\text{clust}}, X_i), \quad (1)$$

where $\text{BIC}_{\text{clust}}(\chi_{\text{clust}}, X_i)$ is the BIC value for the "best" clustering mixture model (i.e., assuming $G \geq 2$) fitted using the features set $\{\chi_{\text{clust}} \cup X_i\}$, whereas $\text{BIC}_{\text{not clust}}(\chi_{\text{clust}}, X_i)$ is the BIC value for no clustering for the same set of variables. The latter can be written as

$$\text{BIC}_{\text{not clust}}(\chi_{\text{clust}}, X_i) = \text{BIC}_{\text{clust}}(\chi_{\text{clust}}) + \text{BIC}_{\text{reg}}(X_i | \chi_{\text{clust}}), \quad (2)$$

i.e., the BIC value for the "best" clustering model fitted using the set χ_{clust} plus the BIC value for the regression of the candidate variable X_i on the variables included in the set χ_{clust}. The difference in BIC score in Equation 1 is an approximation of the log of the Bayes factor comparing the model where the variable under consideration, X_i, is a clustering

variable with the model where the variable is conditionally independent of the clustering. Large, positive values of BIC_{diff} can be taken as an evidence that variable X_i is useful for clustering.

In all clustering models, the "best" model is identified with respect to the number of mixture components (assuming $G \geq 2$) and to model parameterization. In the linear regression model term, X_i can depend on all the variables in χ_{clust}, a subset of them, or none (complete independence). Thus, following the proposal of Maugis *et al.* (2009a), the regression on all the already selected clustering variables is replaced by regression on a subset of them, chosen by a stepwise method. Finally, note that in both Equations 1 and 2 the set of remaining variables, χ_{other}, plays no role.

As described by Raftery and Dean (2006) and Dean (2006), practical implementation of the above methodology requires the use of an algorithm for checking single variables for inclusion/exclusion from the set of selected clustering variables. The package **clustvarsel** implements two different algorithms: a stepwise greedy search algorithm, and a headlong algorithm. Both are based on the concept of a current set of selected clustering variables that expands or contracts at each step of the algorithm. They both also alternate between inclusion and exclusion steps, stopping when neither changes the current set. They are thus algorithms that actually exactly, rather than approximately to within a tolerance.

At each inclusion step, the *stepwise greedy search* algorithm considers each variable not in the current set of selected clustering variables in turn, and assesses the evidence in favor of adding it to the current set. If the evidence is against inclusion for all the variables not in the current set, no change is made. Otherwise, the variable for which the evidence in favor of inclusion is highest is added to the current set. Similarly, at each exclusion step, the algorithm assesses the evidence for removal of each variable in the current set, and removes the variable for which the evidence of removal is highest, provided that this evidence is positive.

The stepwise algorithm can be implemented in a *forward/backward* fashion, i.e., starting from the empty set of clustering variable and then continuing to add or remove features until there is no evidence of further clustering variables. It can also be implemented in a *backward/forward* fashion, i.e., starting from the full set of features as clustering variables and then continuing to remove or add features until there is no evidence of further clustering variables.

The basic idea is similar to stepwise regression and could in principle suffer from the instabilities of stepwise regression discussed by Miller (2002). However, we have not observed this in any of the numerous simulations we have carried out and examples we have analyzed with the method.

Unlike the stepwise greedy algorithm, at each inclusion step the *headlong search* algorithm does not check all the variables not in the current set. Instead it checks the variables one at a time until a variable is found for which the evidence in favor of inclusion exceeds a prespecified threshold. The default threshold is 0, in which case the inclusion step ends whenever a variable is found with positive evidence for inclusion. Any variable for which

the evidence for inclusion is below a prespecified level (the default is a BIC difference of −10) is removed from consideration for the rest of the algorithm. This allows variables that are likely to be irrelevant to be removed early in the algorithm.

Similarly, at each exclusion step, variables are checked one at a time until a variable is found for which the evidence for clustering versus not clustering is below the threshold, at which point that variable is removed and the exclusion step ends. See Badsberg (1992) for further details about the headlong algorithm.

The headlong algorithm involves fewer calculations than the stepwise greedy algorithm, and so is faster, sometimes much faster. The flip side is that it is a greedier algorithm, and so it may find a less good solution.

3. The R package clustvarsel

The **clustvarsel** package can be used to find the (locally) optimal subset of variables with group/cluster information in a dataset with continuous variables. In this section, usage of the main function `clustvarsel` and its arguments is described.

The **clustvarsel** package depends on other packages available on CRAN for model fitting (mclust, Fraley, Raftery, and Scrucca 2017), or for providing some facilities, such as parallelization (**parallel**, R Core Team 2017; **doParallel**, Microsoft Corporation and Weston 2017; **foreach**, Revolution Analytics and Weston 2015a; **iterators**, Revolution Analytics and Weston 2015b), and subset selection in regression models (**BMA**, Raftery, Hoeting, Volinsky, Painter, and Yeung 2017). By loading the package as usual with `library( "clustvarsel")`, it will also take care of making all the other packages available for the current session.

Once the **clustvarsel** package has been loaded, the main function a user needs to invoke is the following:

```
clustvarsel(data, G = 1:9, search = c("greedy", "headlong"),
  direction = c("forward", "backward"), emModels1 = c("E", "V"),
  emModels2 = mclust.options("emModelNames"), samp = FALSE,
  sampsize = round(nrow(data) / 2), hcModel = "VVV", allow.EEE = TRUE,
  forcetwo = TRUE, BIC.diff = 0, BIC.upper = 0, BIC.lower = -10,
  itermax = 100, parallel = FALSE, verbose = interactive())
```

The available arguments are:

`data` A numeric matrix or data frame where rows correspond to observations and columns correspond to variables. Categorical variables are not allowed.

`G` An integer vector specifying the numbers of mixture components (clusters) for which the BIC is to be calculated. The default is `G = 1:9`.

`search` A character vector indicating whether a `"greedy"` or, potentially quicker but less optimal, `"headlong"` algorithm is to be used in the search for clustering variables.

`direction` A character vector indicating the type of search: `"forward"` starts from the empty model and at each step of the algorithm adds/removes a variable until the stopping criterion is satisfied; `"backward"` starts from the model with all the available variables and at each step of the algorithm removes/adds a variable until the stopping criterion is satisfied. For the `"headlong"` search only the `"forward"` algorithm is available.

`emModels1` A vector of character strings indicating the models to be fitted in the expectation maximization (EM) phase of univariate clustering. Possible models are `"E"` and `"V"`, described in `help(mclustModelNames)`.

`emModels2` A vector of character strings indicating the models to be fitted in the EM phase of multivariate clustering. Possible models are described in `help(mclustModelNames)`.

`samp` A logical value indicating whether or not a subset of observations should be used in the hierarchical clustering phase used to get starting values for the EM algorithm.

`sampsize` The number of observations to be used in the hierarchical clustering subset. By default, a random sample of approximately half of the sample size is used.

`hcModel` A character string specifying the model to be used in hierarchical clustering for choosing the starting values used by the EM algorithm. By default, the unconstrained `"VVV"` covariance structure is used.

`allow.EEE` A logical value indicating whether a new clustering will be run with equal within-cluster covariance for hierarchical clustering to get starting values, if the clusterings with variable within-cluster covariance for hierarchical clustering do not produce any viable BIC values.

`forcetwo` A logical value indicating whether at least two variables will be forced to be selected initially, regardless of whether BIC evidence suggests bivariate clustering or not.

`BIC.diff` A numerical value indicating the minimum BIC difference between clustering and no clustering used to accept the inclusion of a variable in the set of clustering variables in a forward step of the greedy search algorithm. Furthermore, minus `BIC.diff` is used to accept the exclusion of a selected variable from the set of clustering variable in a backward step of the greedy search algorithm. Default is 0.

`BIC.upper` A numerical value indicating the minimum BIC difference between clustering and no clustering used to select a clustering variable in the headlong search. Default is 0.

`BIC.lower` A numerical value indicating the level of BIC difference between clustering and no clustering below which a variable will be removed from consideration in the headlong algorithm. Default is −10.

`itermax` An integer value giving the maximum number of iterations (of addition and removal steps) the selected algorithm is allowed to run for.

`parallel` This argument allows to specify if the selected `"greedy"` algorithm should be run sequentially or in parallel. For a single machine with multiple cores, the possible values are:

- **i.** a logical value specifying if parallel computing should be used (`TRUE`) or not (`FALSE`, default) for running the algorithm;
- **ii.** a numerical value which gives the number of cores to employ (by default, this is obtained from function `detectcores` in the **parallel** package);
- **iii.** a character string specifying the type of parallelization to use. The latter depends on the operating system: on Windows only `"snow"` type functionality is available, whereas on Unix/Linux/Mac OS X both `"snow"` and `"multicore"` (default) functionalities are available.

Options (ii) and (iii) imply that the search is performed in parallel, and at the end of the search the cluster is automatically stopped by shutting down the workers. If a cluster of multiple machines is available, the algorithm can be run in parallel using all, or a subset of, the cores available to the machines belonging to the cluster. However, this option requires more work from the user, who needs to set up and register a parallel back end. In this case, the cluster must be explicitly stopped using the function `stopCluster` from the **parallel** package. By default the algorithm is run sequentially.

`verbose` A logical value indicating if information should be provided at each step of the algorithm. By default is set to `TRUE` during interactive sessions, and `FALSE` otherwise.

A basic `clustvarsel` function call needs to input a matrix or data frame containing the data to analyze. Fine tuning is possible by specifying the arguments described above. The following section presents some examples of its usage in practice.

We conclude this section by noting that the initialization of EM is often crucial in fitting mixture models because the likelihood surface tends to have multiple modes. In **mclust** the EM algorithm is initialized using the partitions obtained from model-based hierarchical agglomerative clustering (MBHAC, Banfield and Raftery 1993). MBHAC is convenient because the underlying probabilistic model is shared by both the initialization step and the model fitting step. Furthermore, MBHAC is computationally advantageous because a single run provides the basis for initializing the EM algorithm for any number of components and covariance decompositions. However, problems may arise in presence of coarse data (i.e., discrete data or rounded continuous data) due to the presence of ties. In certain circumstances the final EM solution may even depend on the ordering of the variables. Scrucca and Raftery (2015) have recently proposed a simple approach to overcome these

drawbacks based on data projection via a suitable transformation before applying MBHAC. Once an initial partition is obtained, the EM algorithm is run as usual on the original variables. Thus, by default, **clustvarsel** employs a scaled singular values decomposition (SVD) transformation by setting `mclust.options(hcUse = "SVD")`. This often yields an improved model likelihood and, of particular importance in subset selection, removes any variables ordering effect.

4. Examples

In this section we present some data analysis examples based on simulated data and on well-known real datasets. All calculations, and in particular the system times reported, have been carried out on a iMac with 4 cores i5 Intel CPU running at 2.8 GHz and with 16GB of RAM, unless explicitly indicated otherwise.

4.1. Simulated data

We consider some of the synthetic data examples described in Maugis *et al.* (2009a). Samples were simulated for a 10-dimensional feature vector where only the first two variables provide clustering information. These were generated from a mixture of four Gaussian distributions $\boldsymbol{X}_{[1:2]} \sim N(\boldsymbol{\mu}_k, \boldsymbol{I}_2)$ with $\boldsymbol{\mu}_1 = (-2, -2)$, $\boldsymbol{\mu}_2 = (-2, 2)$, $\boldsymbol{\mu}_3 = -\boldsymbol{\mu}_2$, $\boldsymbol{\mu}_4 = -\boldsymbol{\mu}_1$, and mixing probabilities $\boldsymbol{\pi} = (0.3, 0.2, 0.3, 0.2)$. The remaining eight variables were simulated according to the model $\boldsymbol{X}_{[3:10]} = \boldsymbol{X}_{[1:2]}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$, where $\boldsymbol{\varepsilon} \sim N(\boldsymbol{0}, \boldsymbol{\Omega})$. Different settings for $\boldsymbol{\beta}$ and $\boldsymbol{\Omega}$ define seven different scenarios (see Table 1 in Maugis *et al.* 2009a). These range from independence of clustering variables on the other features (model 1 and 2) to cases of increasing degree of dependence of the irrelevant variables on the clustering ones (model 3 to 7). In the following, we focus only on some of the scenarios. For ease of reading, the values of the parameters for such scenarios are reported in Table 1.

The simulation results were evaluated using the following criteria:

- Variable selection error rate (VSER) to assess *variable selection performance*. VSER is defined as the ratio of the number of errors in selecting (or not selecting) variables to the total number of variables in the set. A perfect recovery of clustering variables gives VSER = 0, while VSER can be no greater than 1.
- Adjusted Rand index (ARI, Hubert and Arabie 1985) to measure *classification accuracy*. A perfect classification gives ARI = 1, whereas ARI = 0 for a random classification.

Table 2 shows the results from a simulation study for the above synthetic data using sample sizes $n = 200$ and $n = 1000$. The methods compared are `MCLUST`, the best Gaussian mixture model (GMM) using the full set of variables, `CLUSTVARSEL[fwd]` and `CLUSTVARSEL[bkw]`, the best GMM using the subset of relevant clustering variables selected by, respectively, the forward/backward and backward/forward greedy search, and `SPARSEKMEANS`, a sparse version of *k*-means algorithm proposed by Witten and Tibshirani (2010) and implemented in the R package **sparcl** (Witten and Tibshirani 2013). Because the last method needs the number of clusters to be fixed in advance, we also included in the comparison versions of the methods based on GMMs with the number of components fixed

at the true number of clusters, i.e., $G = 4$. Finally, note that true subset size is 2, so the optimal VSER should be 0, and the best average ARI value attainable, using the true clustering variables and fixed $G = 4$ components, is about 0.88.

Compared to the performance of `CLUSTVARSEL` reported in Table 1 of Maugis *et al.* (2009a), the new version of the algorithm is able to correctly discard irrelevant variables, both when they are independent of the clustering ones and when they are correlated.

When G is fixed at the true number of clusters, `MCLUST` gives slightly less accurate results for $n = 200$, except in the case of complete independence (scenario 1). `CLUSTVARSEL` provides equivalent accuracy, both if a forward/backward search or a backward/forward search is used. `SPARSEKMEANS` shows results equivalent to greedy search in term of accuracy, but it tends to select (i.e., assigns weights different from zero to) too many variables. Consequently, the VSER of `SPARSEKMEANS` is always worse than that of `CLUSTVARSEL`.

When G is unknown, `MCLUST` often provides inaccurate clustering, in particular when $n = 200$. On the contrary, `CLUSTVARSEL` is generally able to select the true clustering variables (i.e., VSER is near or exactly zero), and also provides very accurate clustering (i.e., ARI is close to 0.88). The only exceptions are scenarios 1 and 4, for the backward/forward search when $n = 200$. In these cases the number of selected variables is slightly larger, which in turn causes a small degradation of clustering accuracy. However, for $n = 1{,}000$ the forward/backward and backward/forward greedy searches are equivalent.

4.2. Crabs data

The crabs dataset in the **MASS** package contains five morphological measurements on 200 specimens of Leptograpsus variegatus crabs recorded on the shore in Western Australia (Campbell and Mahon 1974). Crabs are classified according to their color (blue and orange) and sex, giving four groups. Fifty specimens are available for each combination of color and sex.

```
R > data("crabs", package = "MASS")

R> X <- crabs[, 4:8]

R> Class <- with(crabs, paste(sp, sex, sep = "|"))

R> table(Class)

Class

        B|F  B|M  O|F  O|M
         50   50   50   50
```

First we look at the result obtained using the function `Mclust` from the **mclust** package, with the best model selected by BIC for clustering on all the variables, allowing all possible parameterizations and the number of groups to range over 1 to 5:

```
R> mod1 <- Mclust(X, G = 1:5)

R> summary(mod1)

----------------------------------------------------

Gaussian finite mixture model fitted by EM algorithm

----------------------------------------------------

Mclust EEV (ellipsoidal, equal volume and shape) model with 4 components:

        log.likelihood    n  df        BIC       ICL
             -1241.006  200  68  -2842.298  -2854.29

Clustering table:

         1   2   3   4
        60  55  39  46
```

The estimated maximum a posteriori (MAP) classification is obtained from `mod1$classification`, so a table comparing the estimated and the true classifications, the corresponding misclassification error rate and the adjusted Rand index (ARI), can be obtained as follows:

```
R> table(Class, mod1$classification)

        Class   1   2   3   4
          B|F  49   0   0   1
          B|M  11   0  39   0
          O|F   0   5   0  45
          O|M   0  50   0   0

R> classError(Class, mod1$classification)$errorRate

[1] 0.085

R> adjustedRandIndex(Class, mod1$classification)

[1] 0.793786
```

The algorithm for selecting the variables that are useful for clustering can be run with the following code:

```
R> (out <- clustvarsel(X, G = 1:5))

------------------------------------------------------

Variable selection for Gaussian model-based clustering

Stepwise (forward/backward) greedy search

------------------------------------------------------

        Variable proposed  Type of step   BICclust  Model  G    BICdiff  Decision
                       CW           Add  -1408.710      E  2   -6.21775  Accepted
                       RW           Add  -1908.964    EEV  2  127.38583  Accepted
                       FL           Add  -2357.252    EEV  4   81.24626  Accepted
                       FL        Remove  -1908.964    EEV  2   81.24626  Rejected
                       BD           Add  -2609.777    EEV  4   56.08094  Accepted
                       BD        Remove  -2357.252    EEV  4   56.08094  Rejected
                       CL           Add  -2842.298    EEV  4  -31.07119  Rejected
                       BD        Remove  -2357.252    EEV  4   56.08094  Rejected

Selected subset: CW, RW, FL, BD
```

By default, a greedy forward/backward search is used. The printed output shows the trace of the algorithm: at each step the most important variable is considered (`Variable proposed`) for addition or deletion (`Type of step`) from the set of active clustering variables. The column `BICclust` contains the BIC values for the “best” clustering model (i.e., the first term in the right hand side of Equation 1), followed by the corresponding model abbreviation (`Model`) and number of mixture components (`G`). The last two columns report, respectively, the criterion in Equation 1 (`BICdiff`) and the final decision (`Decision`), which can be either to accept or reject the proposal.

In this example, the final subset contains four out of five morphological features:

```
R> out$subset

CW RW FL BD
 4  2  1  5
```

The same subset is also obtained by using a backward/forward greedy search:

```
R> clustvarsel(X, G = 1:5, direction = “backward”)

------------------------------------------------------
```

```
Variable selection for Gaussian model-based clustering

Stepwise (backward/forward) greedy search

------------------------------------------------------

        Variable proposed  Type of step   BICclust  Model  G    BICdiff  Decision
                       CL        Remove  -2609.777    EEV  4  -31.07119  Accepted
                       BD        Remove  -2357.252    EEV  4   56.08094  Rejected

Selected subset: FL, RW, CW, BD
```

The identified subset can be used for fitting the final clustering model as follows:

```
R> Xs <- X[, out$subset]

R> mod2 <- Mclust(Xs, G = 1:5)

R> summary(mod2)

------------------------------------------------------

Gaussian finite mixture model fitted by EM algorithm

------------------------------------------------------

Mclust EEV (ellipsoidal, equal volume and shape) model with 4 components:

        log.likelihood    n  df        BIC        ICL
             -1180.378  200  47  -2609.777  -2624.892

Clustering table:

         1   2   3   4
        53  60  40  47
```

The accuracy of the clustering obtained on the selected subset of variables is obtained as:

```
R> table(Class, mod2$classification)

        Class   1   2   3   4
          B|F   0  50   0   0
          B|M   0  10  40   0
          O|F   3   0   0  47
          O|M  50   0   0   0
```

```
R> classError(Class, mod2$classification)$errorRate

[1] 0.065

R> adjustedRandIndex(Class, mod2$classification)

[1] 0.8399679
```

Finally, to get an idea of system times required by the subset selection procedure, the following code can be used (note that elapsed times are expressed in seconds):

```
R> library("rbenchmark")

R> benchmark(clustvarsel(X, G = 1:5, verbose = FALSE),

+clustvarsel(X, G = 1:5, direction = "backward", verbose = FALSE),

+columns = c("test", "elapsed"), order = NULL, replications = 1)

                                                        test  Elapsed
1           clustvarsel(X, G = 1:5, verbose = FALSE)    2.998
2    clustvarsel(X, G = 1:5, direction = "backward",    1.551
                                        verbose = FALSE)
```

4.3. Coffee data

Data on twelve chemical constituents of coffee for 43 samples were collected from 29 countries around the world (Streuli 1973). Each coffee sample is either of the Arabica or Robusta variety. The dataset is available in the R package `pgmm` (McNicholas, ElSherbiny, McDaid, and Murphy 2015).

```
R> data("coffee", package = "pgmm")

R> X <- as.matrix(coffee[, 3:14])

R> Class <- factor(coffee$Variety, levels = 1:2,

+  labels = c("Arabica", "Robusta"))

R> table(Class)

Class

   Arabica  Robusta
        36        7

R> mod1 <- Mclust(X)
```

```
R> summary(mod1)

----------------------------------------------------

Gaussian finite mixture model fitted by EM algorithm

----------------------------------------------------

Mclust VEI (diagonal, equal shape) model with 3 components:

        log.likelihood   n  df        BIC        ICL
             -392.9397  43  52  -981.4619  -981.6379

Clustering table:

         1   2  3
        22  14  7
```

Model-based clustering applied to this dataset selects the `VEI` model with 3 components. The clustering table and the corresponding adjusted Rand index (ARI) are the following:

```
R> table(Class, mod1$classification)

        Class       1   2  3
          Arabica  22  14  0
          Robusta   0   0  7

R> adjustedRandIndex(Class, mod1$classification)

[1] 0.3833116
```

The Arabica variety appears to be split into two sub-varieties, whereas the Robusta is correctly identified as a single cluster. As a result, a small value of ARI is obtained.

Now, we may try variable selection to drop irrelevant features, and see if we can improve upon the above model. The following code uses the backward/forward greedy search for variable selection, which by default is performed over all the covariance decomposition models and numbers of mixture components from 1 up to 9:

```
R> (out <- clustvarsel(X, direction = “backward”))

----------------------------------------------------

Variable selection for Gaussian model-based clustering

Stepwise (backward/forward) greedy search
```

```
----------------------------------------------------

          Variable proposed  Type of step   BICclust  Model  G     BICdiff  Decision
              Extract Yield        Remove  -788.3021    VEI  3  -10.930431  Accepted
          Neochlorogenic Acid        Remove  -852.5413    VEI  3   -9.982637  Accepted
             Chlorogenic Acid        Remove  -805.6227    VEI  3  -11.065351  Accepted
              Extract Yield           Add  -999.1101    VEI  3   -9.315106  Rejected
          Isochlorogenic Acid        Remove  -816.8139    VEV  8  -13.958685  Accepted
              Extract Yield           Add  -936.2985    VEV  6   66.325955  Accepted
              Extract Yield        Remove  -816.8139    VEV  8   66.325955  Rejected
          Isochlorogenic Acid           Add  -999.1101    VEI  3  -90.028182  Rejected

Selected subset: Water, Bean Weight, ph Value, Free Acid, Mineral Content,
Fat, Caffine, Trigonelline, Extract Yield
```

Then, the clustering model estimated on the selected subset of variables is:

```
R> mod2 <- Mclust(X[, out$subset])

R> summary(mod2)

----------------------------------------------------

Gaussian finite mixture model fitted by EM algorithm

----------------------------------------------------

Mclust EEI (diagonal, equal volume and shape) model with 3 components:

        log.likelihood   n  df        BIC        ICL
             -443.2269  43  38  -1029.379  -1030.937

Clustering table:

         1   2  3
        22  14  7

R> table(Class, Cluster = mod2$class)

                  Cluster
        Class      1   2  3
          Arabica  22  14  0
          Robusta   0   0  7

R> table(Class, Cluster = mod2$class)
```

```
           Cluster
Class       1  2  3
  Arabica  22 14  0
  Robusta   0  0  7
```

Both the covariance parameterization (EEE) and the number of mixture components (3) used with the selected features subset agree with those from the model using all the variables. The final clustering confirms the structure we already discussed, in particular the two sub-varieties of Arabica coffee.

To show graphically these findings, we may project the data onto a dimension reduced subspace by using the methodology described in Scrucca (2010):

```
R> mod2dr <- MclustDR(mod2)

R> plot(mod2dr, what = "scatterplot", symbols = c("A", "a", "R"))
```

From Figure 1 there is an evident separation between Arabica and Robusta coffee samples along the first direction. Moreover, it seems to confirm the non homogeneous group of Arabica samples, which splits in two sub-varieties along the second direction. Finally, to get an idea of computing time required by the subset selection procedure, the following code can be used:

The system times (in seconds) required by the two stepwise greedy searches can be shown using the following code:

```
R> library("rbenchmark")

R> benchmark(clustvarsel(X, verbose = FALSE),

+  clustvarsel(X, direction = "backward", verbose = FALSE),

+  columns = c("test", "elapsed"), order = NULL, replications = 1)
```

	test	elapsed
1	clustvarsel(X, verbose = FALSE)	8.19
2	clustvarsel(X, direction = "backward", verbose = FALSE)	7.74

4.4. Simulated high-dimensional data

Witten and Tibshirani (2010, Section 3.3.2) discussed an example where five clustering variables are conditionally independent given the cluster memberships, whereas the remaining twenty features are simply independent standard normal variables, also independent from the clustering ones. The first five variables are distributed according to a spherical Gaussian distribution with mean $\boldsymbol{\mu}_1 = (\mu, \mu, \ldots, \mu)$, $\boldsymbol{\mu}_2 = \mathbf{0}$, $\boldsymbol{\mu}_3 = -\boldsymbol{\mu}_1$, where $\mu = 1.7$,

and common unit standard deviation. We replicated this experiment (denoted by WT) with varying sample sizes (n_g cases for each group) and for a set of different techniques.

The algorithms we consider in the comparison are `SPARSEKMEANS`, and `CLUSTVARSEL` using both the forward/backward and the backward/forward greedy search. Furthermore, `K-MEANS`, and `MCLUST` using either the first five variables and all the variables have been included as benchmarks. For `MCLUST` and `CLUSTVARSEL` models the EII parameterization is used both at the hierarchical initialization step and for mixture modeling.

Table 3 reports the variable selection error rate (VSER) and the classification error rate (CER). As already mentioned in Section 4.1, the VSER is defined as the ratio of the number of errors in selecting (or not selecting) variables with respect to the total number of variables considered. The CER between two partitions, which is equivalent to one minus the Rand index (Rand 1971), is equal to 0 in the case of perfect agreement, and becomes larger for increasing disagreement (for a formal definition see Witten and Tibshirani 2010). Smaller values of both VSER and CER are better. These two measures have been chosen for the purpose of comparison with the results in Witten and Tibshirani (2010, Table 4) and Celeux, Martin-Magniette, Maugis-Rabusseau, and Raftery (2014, Section 3.1).

The model with uniformly better performance in terms of classification error is the `MCLUST` model with the first five variables, i.e., the model which most resembles the data generation mechanism. However, this model is not available when we do not know the clustering variables, as we are assuming here. `MCLUST` using all the variables has quite good accuracy and improves as group sample size increases. On the contrary, `K-MEANS` on all the variables shows a constant larger classification error. `SPARSEKMEANS` performs well in term of accuracy, but the number of irrelevant variables selected increases with group sample size, and all the features tend to be selected for $n_g = 50$ (VSER is almost equal to 20/25 = 0.8). `CLUSTVARSEL` using backward/forward greedy search shows good accuracy as well, and improves as group sample size gets larger. The variable selection error is the smallest when n_g gets larger. Thus, for increasing sample size it converges to the true subset size. Note that forward/backward greedy search is clearly less accurate than the backward/forward search, with the VSER which is also larger, so the performance is overall worst than that of backward/forward search.

5. Adjustments for speeding up the algorithm

5.1. Sub-sampling at hierarchical initialization step

As mentioned in Section 3, the EM algorithm is initialized in **mclust** using the partitions obtained from model-based agglomerative hierarchical clustering. Efficient numerical algorithms for approximately maximizing the classification likelihood with multivariate normal models have been discussed by Fraley (1998). However, for datasets having a large number of observations this step can be computationally expensive.

When the number of observations is large, we may allow `clustvarsel` to use only a subset of the observations at the model-based hierarchical stage of clustering, to speed up the

algorithm. This is easily done by setting the argument `samp = TRUE`, and by specifying the number of observations to be used in the hierarchical clustering subset with `sampsize`.

Consider the following simulation scheme which constructs a medium sized dataset on five dimensions. Only the first two variables contain clustering information, the third is highly correlated with the first one, whereas the remaining features are simply noise variables.

```
R> set.seed(5)

R> library("MASS")

R> library("rbenchmark")

R> n <- 1000

R> pro <- 0.5

R> mu1 <- c(0, 0)

R> mu2 <- c(3, 3)

R> sigma1 <- matrix(c(1, 0.5, 0.5, 1), 2, 2)

R> sigma2 <- matrix(c(1.5, -0.7, -0.7, 1.5), 2, 2)

R> X <- matrix(0, n, 5, dimnames = list(NULL, paste0("X", 1:5)))

R> u <- runif(n)

R> Class <- ifelse(u < pro, 1, 2)

R> X[u < pro, 1:2] <- mvrnorm(sum(u < pro), mu = mu1, Sigma = sigma1)

R> X[u >= pro, 1:2] <- mvrnorm(sum(u >= pro), mu = mu2, Sigma = sigma2)

R> X[, 3] <- X[, 1] + rnorm(n)

R> X[, 4] <- rnorm(n, mean = 1.5, sd = 2)

R> X[, 5] <- rnorm(n, mean = 2, sd = 1)

R> clPairs(X, Class, gap = 0.2)
```

We may compare the procedure which uses sampling at the hierarchical stage with the default call to `clustvarsel`, both in terms of computing time, using the function `system.time`, and in terms of clustering accuracy.

```
R> benchmark(out1 <- clustvarsel(X, G = 1:5, verbose = FALSE),
```

```
+out2 <- clustvarsel(X, G = 1:5, samp = TRUE, sampsize = 200,

+verbose = FALSE), columns = c("test", "elapsed", "relative"),

+order = NULL, replications = 1)
```

	test	elapsed	relative
1	out1 <- clustvarsel(X, G = 1:5)	5.560	1.798
2	out2 <- clustvarsel(X, G = 1:5, samp = TRUE, sampsize = 200)	3.093	1.000

Thus, by using sub-sampling for the initial hierarchical clustering we obtain a 1.65-fold speedup over the original with the same accuracy:

```
R> out1$subset

  X2  X1
   2   1

R> out2$subset

  X2  X1
   2   1
```

To investigate the effect of sampling as the number of observations increase we conducted a small simulation study by replicating the above simulation setting with different sample sizes and fixed size at 200 observations for choosing the initial starting points. Figure 3 shows the results averaged over 10 replications. Panel (a) reports the computing time required as the sample size grows, whereas panel (b) shows the relative gain from using a subset of observations at the initial hierarchical stage. As can be seen, efficiency improves roughly exponentially as the number of observations increases, with sampling being about 50 times faster at 10, 000 cases. As the system time required increases linearly for sampling, when no sampling is used at the initial stage the time required increases approximately exponentially. Furthermore, in all the replications the first two variables have been selected by both methods. Hence, the improvement in terms of computational efficiency has not caused any deterioration in terms of accuracy.

5.2. Headlong search

When a dataset contains a large number of variables we may find that using the headlong search algorithm option (`search = "headlong"`) is faster than the default greedy search. To show an example we simulated a dataset analogous to the previous one for the clustering variables, then six more irrelevant variables were added, some correlated with the clustering ones, some independent and some correlated among themselves.

```
R> set.seed(7)
```

```
R> library("MASS")
R> library("rbenchmark")
R> n <- 400
R> pro <- 0.5
R> mu1 <- c(0, 0)
R> mu2 <- c(3, 3)
R> sigma1 <- matrix(c(1, 0.5, 0.5, 1), 2, 2)
R> sigma2 <- matrix(c(1.5, -0.7, -0.7, 1.5), 2, 2)
R> X <- matrix(0, n, 10, dimnames = list(NULL, paste0("X", 1:10)))
R> u <- runif(n)
R> Class <- ifelse(u < pro, 1, 2)
R> X[u < pro, 1:2] <- mvrnorm(sum(u < pro), mu = mu1, Sigma = sigma1)
R> X[u >= pro, 1:2] <- mvrnorm(sum(u >= pro), mu = mu2, Sigma = sigma2)
R> X[, 3] <- X[, 1] + rnorm(n)
R> X[, 4] <- X[, 2] + rnorm(n)
R> X[, 5] <- rnorm(n, mean = 1.5, sd = 2)
R> X[, 6] <- rnorm(n, mean = 2, sd = 1)
R> X[, 7:8] <- mvrnorm(n, mu = mu1, Sigma = sigma1)
R> X[, 9:10] <- mvrnorm(n, mu = mu2, Sigma = sigma2)
```

Then, we may compare the time required by using the greedy method and using the headlong method:

```
R> benchmark(out1 <- clustvarsel(X, G = 1:5, verbose = FALSE),
+out2 <- clustvarsel(X, G = 1:5, search = "headlong", verbose = FALSE),
+columns = c("test", "elapsed", "relative"),
+order = NULL, replications = 1)
```

```
                                                    test  elapsed  relative
1                 out1 <- clustvarsel(X, G = 1:5)    4.913    1.707
2 out2 <- clustvarsel(X, G = 1:5, search = “headlong”)    2.878    1.000
```

In situations where there are many observations and a large number of variables, sub-sampling at the hierarchical initialization step and the headlong search can be used concurrently to improve computational efficiency. A simulation study was conducted by replicating the previous simulation scheme with different sample sizes. The methods compared are greedy and headlong searches, without and with sampling using `sampize = 200`. The results averaged over 10 replications are shown in Figure 4. Without sampling, headlong search is faster than greedy search with a constant speedup factor of about 1.7. The use of sampling at the initial hierarchical stage enables us to achieve an exponential relative gain as the sample size increases for both type of searches. Note that also in this case, the headlong search maintains an approximate 1.7-fold advantage compared to the greedy search.

The speed/optimally tradeoff in a headlong search can be changed by increasing or decreasing the different levels, e.g., by setting the upper level to 10 instead of 0 we would require a variable to have stronger evidence of clustering before it is included, and by setting the lower level to 0 we would remove variables that at any stage have evidence of clustering weaker by any amount than evidence against clustering.

5.3. Parallel computing

Parallel computing is a form of computation in which the required calculations are performed simultaneously, either on a single multi-core processors machine or on a cluster of multiple computers.

Direct support of parallelism in R is available since version 2.14.0 (released in October 2011) through the package **parallel** (R Core Team 2017). This is essentially a merger of the **multicore** package (Urbanek 2011) and the **snow** package (Tierney, Rossini, Li, and Sevcikova 2016). The `multicore` functionality supports parallelism via forking, which is a concept from POSIX operating systems, so it is available on all R platforms except Windows. In contrast, **snow** supports different transport mechanisms (e.g., socket connections) to communicate between the master and the workers, and it is available on all operating systems. Other approaches to parallel computing in R are available as described in McCallum and Weston (2011). For an extensive list of packages see the CRAN task view on *High-Performance and Parallel Computing with R* (Eddelbuettel 2017).

The greedy search discussed in Section 2 constitutes an embarrassingly parallel problem, i.e., one for which little or no effort is required to separate the problem into a number of parallel tasks. Essentially, the sequential evaluation of candidate variables for inclusion or exclusion, which is the most time consuming task, can be done in parallel. For the actual implementation in **clustvarsel** we used the **doParallel** package (Microsoft Corporation and Weston 2017), a “parallel backend” which acts as an interface between the **foreach** package (Revolution Analytics and Weston 2015a; Kane, Emerson, and Weston 2013) and the

parallel package. Essentially, it provides a mechanism needed to execute for-loops in parallel.

To specify if parallel computing should be used in the evaluation of the BIC_{diff} criterion in Equation 1, the optional argument `parallel` must be set to `TRUE` in the `clustvarsel` function call. In this case all the available cores, as returned by the `detectCores` function, are used. A numeric value specifying the number of cores to employ can also be specified in the optional argument `parallel`. Finally, the parallelization functionality depends on system OS: on Windows, only **snow** type functionality is available, whereas on Unix/Linux/Mac OSX, both **snow** and **multicore** (default) functionalities are available.

As an example, consider a sample of $n = 200$ observations generated according to the simulation scheme described in Section 5.2. We may compare the sequential greedy backward/forward search with a parallel version of the algorithm with the default maximum cores available and by specifying 2 cores:

```
R> benchmark(clustvarsel(X, G = 1:9, direction = “backward”, verbose =
FALSE),

+clustvarsel(X, G = 1:9, direction = “backward”, parallel = TRUE, verbose

+= FALSE), clustvarsel(X, G = 1:9, direction = “backward”, parallel = 2,

+verbose = FALSE), columns = c(“test”, “elapsed”), order = NULL,

+replications = 1)
```

	test	elapsed
1	clustvarsel(X, G = 1:9, direction = “backward”)	52.12
2	clustvarsel(X, G = 1:9, direction = “backward”, parallel = TRUE)	17.46
3	clustvarsel(X, G = 1:9, direction = “backward”, parallel = 2)	28.64

In this case, the execution time is reduced to about a third (52.12/17.46 = 2.985) using 4 cores, whereas with 2 cores a speedup factor of 52.12/28.64 = 1.82 is obtained.

By using a machine with P processors instead of just one, we would like to obtain an increase in calculation speed of P times. As shown above, this is not the case because in the implementation of a parallel algorithm there are some inherent non-parallelizable parts and communication costs between tasks (Nakano 2012). Amdahl's Law (Amdahl 1967) is often used in parallel computing to predict the theoretical maximum speedup when using multiple processors. If f is the fraction of non-parallelizable task, i.e., the part of the algorithm that is strictly serial, and P is the number of processors in use, then the maximum speedup achievable on a parallel computing platform is given by

$$S_P = \frac{1}{f + (1 - f)/P}. \quad (3)$$

In the limit, the above ratio converges to $S_{\max} = 1/f$, which represents the maximum increase of speed achievable in theory, i.e., by a machine with an infinite number of processors.

To investigate the performance of our parallel algorithm implementation, we conducted a small simulation study using the above simulation setting for increasing numbers of cores. The study was performed on a 18 cores Intel Xeon E5-2697 v4 running at 2.30GHz and with 128GB of RAM.

Figure 5 shows the results averaged over 10 replications. The points represent the observed speedup factor (obtained as $s_P = t_1/t_P$, where t_P is the execution time using P cores) for running the greedy backward/forward algorithm with up to 10 cores. The curve represents the Amdahl's Law (3) with f estimated by non-linear least squares. It turns out that the estimated fraction of strictly sequential part in the backward/forward search for variable selection is $f = 0.13$, which yields a maximum speedup of about $S_{\max} = 7.6$.

6. Conclusions and future work

This paper has presented the R package **clustvarsel** which provides a convenient set of tools for variable selection in model-based clustering using a finite mixture of Gaussian densities. Stepwise greedy search and headlong algorithm are implemented in order to find the (locally) optimal subset of variables with cluster information. The computational burden of such algorithms can be decreased by some ad hoc modifications in the algorithms, or via the use of parallel computation as implemented in the package. Examples illustrating the use of the package in practical applications have been presented.

Given the vast solution space, other optimization techniques could be usefully employed. For instance, the use of genetic algorithms as described in Scrucca (2016) will be included in a future release of the package.

The present methodology is restricted to continuous data modeled by mixtures of Gaussian distributions. However, an analogous method was proposed by Dean and Raftery (2010) for multivariate discrete data in the context of latent class analysis models, with improvements recently proposed by Fop, Smart, and Murphy (2017). This methodology could also be included in this or another package.

Acknowledgments

The authors acknowledge the CINECA award under the ISCRA initiative (http://www.hpc.cineca.it/services/iscra/) for the availability of high performance computing resources and support. Adrian E. Raftery and Luca Scrucca were supported by NIH grants R01 HD054511, R01 HD070936 and U54 HL127624.

References

Amdahl GM. Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities. AFIPS Conference Proceedings. 1967; 30:483–485. DOI: 10.1145/1465482.1465560

Andrews JL, McNicholas PD. ***vscc***: Variable Selection for Clustering and Classification. R package version 0.2. 2013. URL https://CRAN.R-project.org/package=vscc

Andrews JL, McNicholas PD. Variable Selection for Clustering and Classification. Journal of Classification. 2014; 31(2):136–153. DOI: 10.1007/s00357-013-9139-2

Badsberg JH. Model Search in Contingency Tables by **CoCo**. In: Dodge Y, Whittaker J, editorsComputational Statistics. Vol. 1. 1992. 251–256.

Banfield JD, Raftery AE. Model-Based Gaussian and Non-Gaussian Clustering. Biometrics. 1993; 49(3):803–821. DOI: 10.2307/2532201

Campbell NA, Mahon RJ. A Multivariate Study of Variation in Two Species of Rock Crab of Genus Leptograpsus. Australian Journal of Zoology. 1974; 22(3):417–425. DOI: 10.1071/zo9740417

Celeux G, Govaert G. Gaussian Parsimonious Clustering Models. Pattern Recognition. 1995; 28(5): 781–793. DOI: 10.1016/0031-3203(94)00125-6

Celeux G, Martin-Magniette ML, Maugis-Rabusseau C, Raftery AE. Comparing Model Selection and Regularization Approaches to Variable Selection in Model-Based Clustering. Journal de la Société Française de Statistique. 2014; 155(2):57–71. [PubMed: 25279246]

Dean N. The Variable Selection for Model-Based Clustering (clustvarsel) Package. Vignette for clustvarsel version 1.0. URL https://CRAN.R-project.org/src/contrib/Archive/clustvarsel/clustvarsel_1.0.tar.gz

Dean N, Raftery AE. Latent Class Analysis Variable Selection. Annals of the Institute of Statistical Mathematics. 2010; 62:11–35. DOI: 10.1007/s10463-009-0258-9 [PubMed: 20827439]

Dean N, Raftery AE, Scrucca L. ***clustvarsel***: Variable Selection for Model-Based Clustering. R package version 2.3.1. 2017. URL https://CRAN.R-project.org/package=clustvarsel

Dia F, Martin-Magniette ML, Maugis C. **SelvarClust** software. 2009a. URL http://www.math.univ-toulouse.fr/~maugis/SelvarClustHomepage.html

Dia F, Martin-Magniette ML, Maugis C. **SelvarClustIndep** software. 2009b. URL http://www.math.univ-toulouse.fr/~maugis/SelvarClustIndepHomepage.html

Eddelbuettel D. CRAN Task View: High-Performance and Parallel Computing with R. Version 2017-11-10. 2017. URL https://CRAN.R-project.org/view=HighPerformanceComputing

Fop M, Smart K, Murphy TB. Variable Selection for Latent Class Analysis with Application to Low Back Pain Diagnosis. The Annals of Applied Statistics. 2017; 11(4):2080–2110. DOI: 10.1214/17-AOAS1061

Fraley C. Algorithms for Model-Based Gaussian Hierarchical Clustering. SIAM Journal on Scientific Computing. 1998; 20(1):270–281. DOI: 10.1137/s1064827596311451

Fraley C, Raftery AE. Model-Based Clustering, Discriminant Analysis, and Density Estimation. Journal of the American Statistical Association. 2002; 97:611–631. DOI: 10.1198/016214502760047131

Fraley C, Raftery AE, Murphy TB, Scrucca L. Technical Report 597. Department of Statistics, University of Washington; 2012. **mclust** Version 4 for R: Normal Mixture Modeling for Model-Based Clustering, Classification, and Density Estimation. URL http://www.stat.washington.edu/research/reports/2012/tr597.pdf

Fraley C, Raftery AE, Scrucca L. ***mclust***: Normal Mixture Modeling for Model-Based Clustering, Classification, and Density Estimation. R package version 5.4. 2017. URL https://CRAN.R-project.org/package=mclust

Hubert L, Arabie P. Comparing Partitions. Journal of Classification. 1985; 2(1):193–218. DOI: 10.1007/bf01908075

Kane MJ, Emerson J, Weston S. Scalable Strategies for Computing with Massive Data. Journal of Statistical Software. 2013; 55(14):1–19. DOI: 10.18637/jss.v055.i14

Kusnierczyk W. rbenchmark: Benchmarking routine for R. R package version 1.0.0. 2012. URL https://CRAN.R-project.org/package=rbenchmark

Maugis C, Celeux G, Martin-Magniette ML. Selection for Clustering with Gaussian Mixture Models. Biometrics. 2009a; 65(3):701–709. DOI: 10.1111/j.1541-0420.2008.01160.x [PubMed: 19210744]

Maugis C, Celeux G, Martin-Magniette ML. Variable Selection in Model-Based Clustering: A General Variable Role Modeling. Computational Statistics & Data Analysis. 2009b; 53(11):3872–3882. DOI: 10.1016/j.csda.2009.04.013

McCallum E, Weston S. Parallel R. O'Reilly Media; Sebastobol: 2011.

McLachlan GJ, Peel D. Finite Mixture Models. John Wiley & Sons; New York: 2000.

McNicholas PD, ElSherbiny A, McDaid AF, Murphy TB. ***pgmm***: Parsimonious Gaussian Mixture Models. R package version 1.2. 2015. URL https://CRAN.R-project.org/package=pgmm

Weston S. Microsoft Corporation. ***doParallel***: Foreach Parallel Adaptor for the ***parallel*** Package. R package version 1.0.11. 2017. URL https://CRAN.R-project.org/package=doParallel

Miller AJ. Subset Selection in Regression. 2nd. Chapman & Hall/CRC; London: 2002.

Nakano J. Parallel Computing Techniques. In: Gentle JE, Härdle WK, Mori Y, editorsHandbook of Computational Statistics. 2nd. Springer-Verlag; Berlin: 2012. 243–271.

Raftery A, Hoeting J, Volinsky C, Painter I, Yeung KY. ***BMA***: Bayesian Model Averaging. R package version 3.18.7. 2017. URL https://CRAN.R-project.org/package=BMA

Raftery AE, Dean N. Variable Selection for Model-Based Clustering. Journal of the American Statistical Association. 2006; 101(473):168–178. DOI: 10.1198/016214506000000113

Rand W. Objective Criteria for the Evaluation of Clustering Methods. Journal of the American Statistical Association. 1971; 66(336):846–850. DOI: 10.1080/01621459.1971.10482356

R Core Team. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing; Vienna, Austria: 2017. URL https://www.R-project.org/

Weston S. Revolution Analytics. ***foreach***: Provides Foreach Looping Construct for R. R package version 1.4.3. 2015a. URL https://CRAN.R-project.org/package=foreach

Weston S. Revolution Analytics. ***iterators***: Provides Iterator Construct for R. R package version 1.0.8. 2015b. URL https://CRAN.R-project.org/package=iterators

Scrucca L. Dimension Reduction for Model-Based Clustering. Statistics and Computing. 2010; 20(4): 471–484. DOI: 10.1007/s11222-009-9138-7

Scrucca L. Genetic Algorithms for Subset Selection in Model-Based Clustering. In: Celebi ME, Aydin K, editorsUnsupervised Learning Algorithms. Springer-Verlag; Cham: 2016. 55–70.

Scrucca L, Fop M, Murphy TB, Raftery AE. mclust 5: Clustering, Classification and Density Estimation Using Gaussian Finite Mixture Models. The R Journal. 2016; 8(1):289–317. [PubMed: 27818791]

Scrucca L, Raftery AE. Improved Initialisation of Model-Based Clustering Using Gaussian Hierarchical Partitions. Advances in Data Analysis and Classification. 2015; 4(9):447–460. DOI: 10.1007/s11634-015-0220-z

Sedki M, Celeux G, Maugis-Rabusseau C. ***SelvarMix***: Regularization for Variable Selection in Model-Based Clustering and Discriminant Analysis. R package version 1.2.1. 2017. URL https://CRAN.R-project.org/package=SelvarMix

Streuli H. Association Scientifique International Du Cafe, 6th International Colloquium on Coffee Chemistry. Bogata, Columbia: 1973. Der Heutige Stand Der Kaffeechemie; 61–72.

Tierney L, Rossini AJ, Li N, Sevcikova H. ***snow***: Simple Network of Workstations. R package version 0.4-2. 2016. URL https://CRAN.R-project.org/package=snow

Urbanek S. ***multicore***: Parallel Processing of R Code on Machines with Multiple Cores or CPUs. R package version 0.1-7. 2011. URL https://CRAN.R-project.org/package=multicore

Witten DM, Tibshirani R. A Framework for Feature Selection in Clustering. Journal of the American Statistical Association. 2010; 105(490):713–726. DOI: 10.1198/jasa.2010.tm09415 [PubMed: 20811510]

Witten DM, Tibshirani R. ***sparcl***: Perform Sparse Hierarchical Clustering and Sparse K-Means Clustering. R package version 1.0.3. 2013. URL https://CRAN.R-project.org/package=sparcl

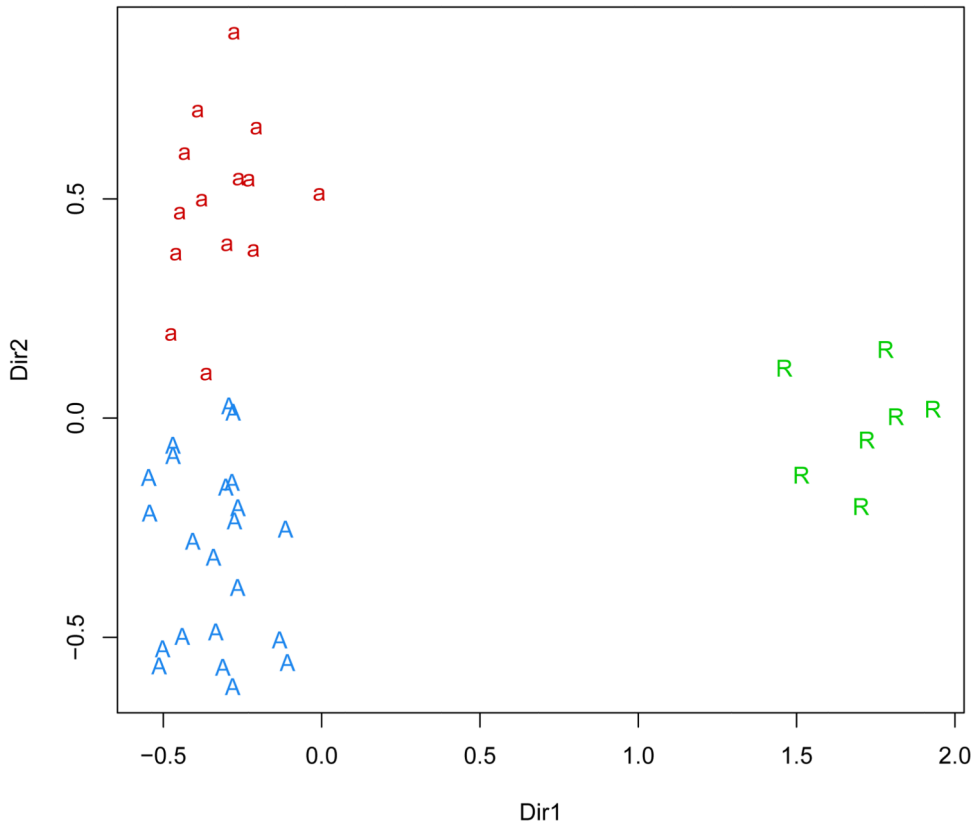


Figure 1.
Projection of coffee data samples marked according to the clustering obtained from the variables selected using the forward/backward greedy search. The symbol R indicates Robusta coffees, A and a the sub-varieties of Arabica coffees.

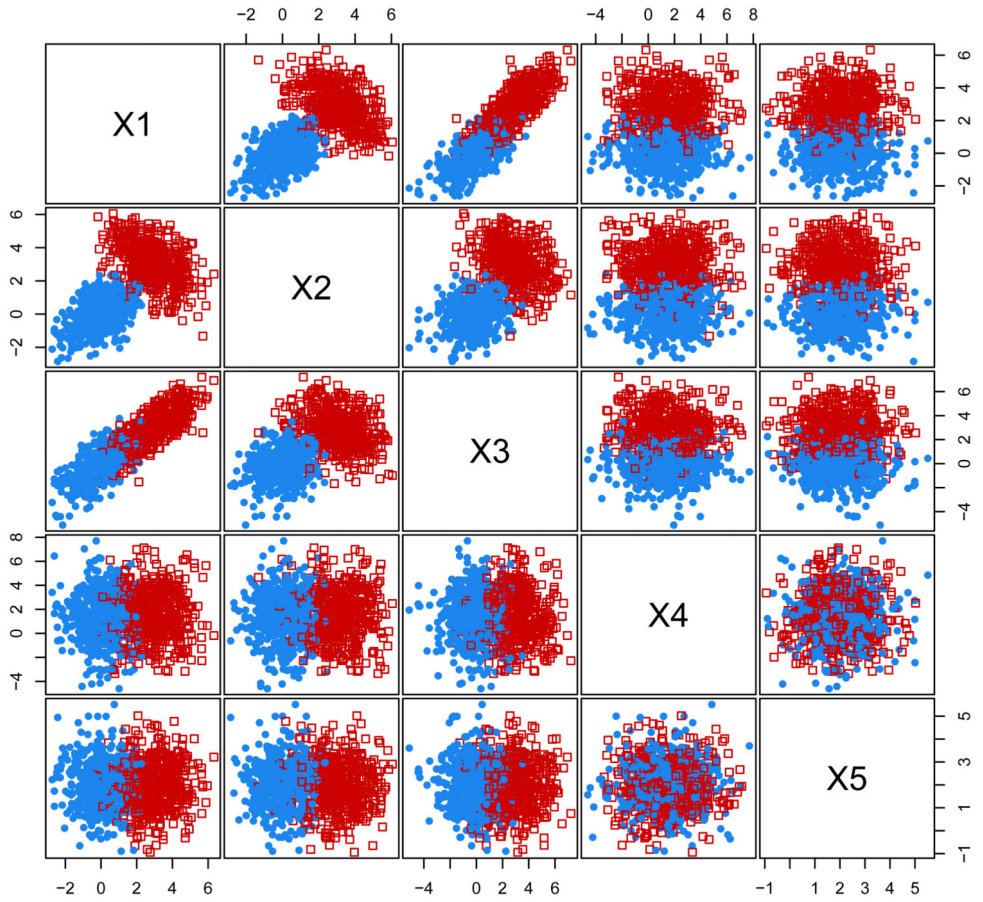
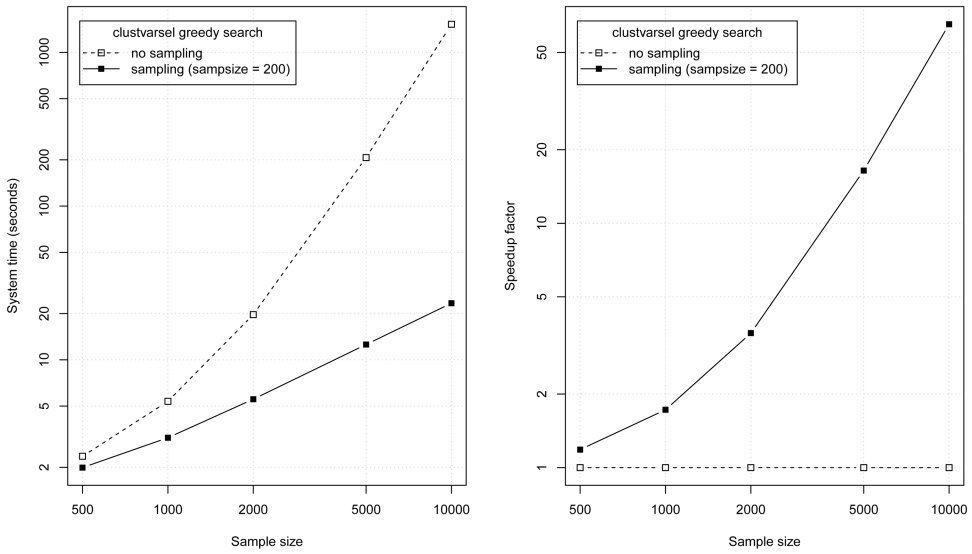


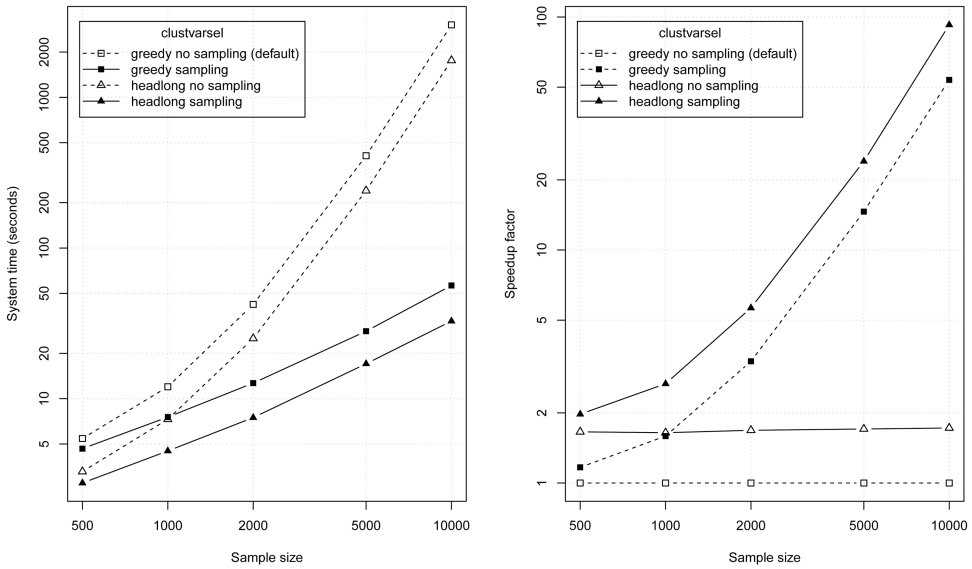
Figure 2.
Scatterplot matrix of simulated data with points marked according to the known groups.



(a) system time vs sample size (b) speedup vs sample size

Figure 3.

Comparison of computing time vs sample size. Panel (a) shows the average over 10 replications for `clustvarsel` using sub-sampling with fixed size at 200 observations, and no sampling. Panel (b) shows the speedup factor, calculated as the ratio of execution time for the greedy search with no sampling to the system times for each examined strategy. All axes are on logarithmic scale.



(a) system time vs sample size (b) relative gain of sub-sampling vs sample size

Figure 4.

Comparison of computing time vs sample size. Panel (a) shows the average over 10 replications for `clustvarsel` using `search` = `"greedy"` with and without sampling, and `search` = `"headlong"` with and without sampling. A fixed value `sampsize` = 200 is used throughout. Panel (b) shows the speedup factor, calculated as the ratio of execution time for the greedy search with no sampling to the system times for each examined strategy. All axes are on logarithmic scale.

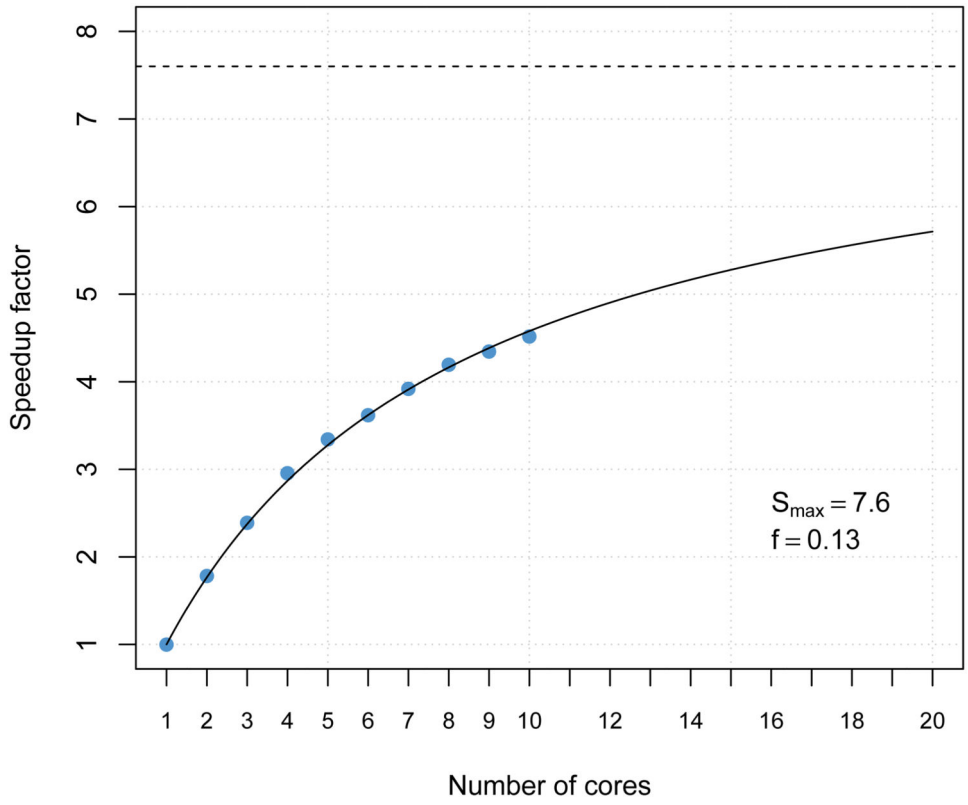


Figure 5.

Graph of speedup factor vs the number of cores employed in the parallel algorithm for greedy backward/forward subset selection in model-based clustering. The fraction of non-parallelizable task is estimated as $f = 0.13$, and, by Amdahl's Law, this gives a maximum speedup achievable by parallelization of around 7.6× the sequential algorithm (dashed horizontal line).

Table 1

Parameter settings for the scenarios used to generated synthetic data: $\boldsymbol{\beta}$ defines the correlation of irrelevant variables on clustering variables, whereas $\boldsymbol{\Omega}$ is the covariance structure of the noise component. $\mathbf{0}_p$ indicates the ($2 \times p$) matrix of zeroes, and $\boldsymbol{I}_p$ the ($p \times p$) identity matrix.

Scenario	Parameters	Scenario	Parameters
Model 1	$\boldsymbol{\beta} = \mathbf{0}_8$	*Model 5*	$\beta = \begin{pmatrix} 0.5 & 0 & 2 & 0 & \\ 0 & 1 & 0 & 3 & \mathbf{0}_4 \end{pmatrix}$
	$\boldsymbol{\Omega} = \boldsymbol{I}_8$		$\boldsymbol{\Omega} = \text{diag}(\boldsymbol{I}_2, 0.5\boldsymbol{I}_2, \boldsymbol{I}_4)$
Model 4	$\beta = \begin{pmatrix} 0.5 & 0 & \\ 0 & 1 & \mathbf{0}_6 \end{pmatrix}$	*Model 7*	$\beta = \begin{pmatrix} 0.5 & 0 & 2 & 0 & 2 & 0.5 & 2 & 0 \\ 0 & 1 & 0 & 3 & 0.5 & 1 & 0 & 3 \end{pmatrix}$
	$\boldsymbol{\Omega} = \boldsymbol{I}_8$		$\boldsymbol{\Omega} = \text{diag}(\boldsymbol{I}_2, 0.5\boldsymbol{I}_4, \boldsymbol{I}_2)$

Table 2

Average values based on 100 simulation runs for some of the models in Maugis *et al.* (2009a), where only two (out of ten) variables are relevant for clustering. The first four models use $G = 4$ fixed number of clusters. `CLUSTVARSEL[fwd]` uses the forward/backward greedy search, whereas `CLUSTVARSEL[bkw]` employs the backward/forward greedy search. True subset size is 2. Smaller values of VSER and larger values of ARI are better. Values within one standard error from the best are shown in bold for each experiment and each criterion. Standard errors for VSER and ARI are all ≤ 0.030.

Model	Subset size	VSER	ARI	Subset size	VSER	ARI
Scenario 1	$n = 200$			$n = 1000$		
`MCLUST,G=4`	10.00	0.800	0.845	10.00	0.800	0.869
`SPARSEKMEANS,G=4`	9.92	0.792	**0.881**	9.27	0.727	0.884
`CLUSTVARSEL[fwd],G=4`	2.01	**0.001**	**0.882**	2.02	**0.002**	**0.887**
`CLUSTVARSEL[bkw],G=4`	2.10	0.010	**0.882**	2.25	0.025	**0.887**
`MCLUST`	10.00	0.800	0.780	10.00	0.800	0.883
`CLUSTVARSEL[fwd]`	2.01	**0.001**	**0.882**	2.01	0.002	**0.887**
`CLUSTVARSEL[bkw]`	4.03	0.311	0.658	2.25	0.025	**0.887**
Scenario 4	$n = 200$			$n = 1000$		
`MCLUST,G=4`	10.00	0.800	0.799	10.00	0.800	0.813
`SPARSEKMEANS,G=4`	9.93	0.793	0.829	9.28	0.728	0.843
`CLUSTVARSEL[fwd],G=4`	2.03	**0.005**	**0.872**	2.00	**0.000**	**0.888**
`CLUSTVARSEL[bkw],G=4`	2.18	0.018	**0.877**	2.22	0.022	**0.888**
`MCLUST`	10.00	0.800	0.640	10.00	0.800	0.864
`CLUSTVARSEL[fwd]`	2.03	**0.005**	**0.872**	2.00	**0.000**	**0.888**
`CLUSTVARSEL[bkw]`	3.13	0.185	0.732	2.25	0.025	**0.888**
Scenario 5	$n = 200$			$n = 1000$		
`MCLUST,G=4`	10.00	0.800	0.811	10.00	0.809	0.879
`SPARSEKMEANS,G=4`	9.35	0.735	0.815	7.00	0.500	0.855
`CLUSTVARSEL[fwd],G=4`	2.00	**0.010**	**0.888**	2.00	**0.000**	**0.884**

Model	Subset size	VSER	ARI	Subset size	VSER	ARI
CLUSTVARSEL[bkw], G=4	2.00	**0.014**	**0.887**	2.00	**0.000**	**0.884**
MCLUST	10.00	0.800	0.455	10.00	0.800	0.879
CLUSTVARSEL[fwd]	1.99	**0.013**	**0.884**	2.00	**0.000**	**0.884**
CLUSTVARSEL[bkw]	2.04	**0.014**	**0.888**	2.04	0.004	**0.884**
Scenario 7	*n* = 200			*n* = 1000		
MCLUST, G=4	10.00	0.800	0.848	10.00	0.800	0.878
SPARSEKMEANS, G=4	9.00	0.704	0.857	9.00	0.700	0.865
CLUSTVARSEL[fwd], G=4	2.00	0.032	**0.878**	2.00	**0.000**	**0.883**
CLUSTVARSEL[bkw], G=4	2.00	0.032	**0.878**	2.00	**0.000**	**0.883**
MCLUST	10.00	0.800	0.469	10.00	0.800	0.878
CLUSTVARSEL[fwd]	2.00	0.032	**0.878**	2.00	**0.000**	**0.883**
CLUSTVARSEL[bkw]	2.00	**0.014**	**0.879**	2.00	**0.000**	**0.883**

Table 3

Average values based on 100 simulation runs for the WT simulation scheme with group sample size n_g. All models assume the number of clusters $G = 3$ to be known. For all the `mclust` models, including those fitted by the variables subset selection algorithms, the EII parameterization was used for both the hierarchical initialization and the mixture modeling. `clustvarsel`[`fwd`] uses the forward/backward greedy search, whereas `clustvarsel`[`bkw`] uses the backward/forward greedy search. Smaller values of both VSER and CER are better. Values within one standard error from the best are shown in bold for each experiment and each criterion. Standard errors are all < 0.1.

	VSER			CER		
Model	$n_g = 10$	$n_g = 20$	$n_g = 50$	$n_g = 10$	$n_g = 20$	$n_g = 50$
`K-MEANS`				0.262	0.242	0.218
`MCLUST`[$X_1, \ldots, X_5$]				0.073	0.055	0.053
`MCLUST`[$X_1, \ldots, X_{25}$]				0.218	0.128	0.063
`SPARSEKMEANS`	**0.138**	0.331	0.799	**0.127**	**0.063**	**0.054**
`CLUSTVARSEL[fwd]`	0.264	0.274	0.114	0.343	0.347	0.151
`CLUSTVARSEL[bkw]`	0.222	**0.086**	**0.033**	0.181	**0.086**	**0.054**

J Stat Softw. Author manuscript; available in PMC 2018 November 16.