Guide on performing feature selection with the R package MXM

Michail Tsagris

Abstract

MXM is a flexible R package offering many feature selection algorithms for predictive or diagnostic models. *MXM* has a unique advantage over other packages; it allows for a long and comprehensive list of types of target variable to be modeled; continuous, percentages, time to event (survival), binary, nominal, ordinal, clustered, counts and left censored to name a few. In this paper we briefly discuss some of these algorithms, the different types of data they can handle, and the relevant functions along with their input and output arguments.

1 Introduction

Given a target (response or dependent) variable Y of n measurements and a set X of p features (predictor or independent variables) the problem of feature (or variable) selection is to identify the minimal set of features with the highest predictive performance. In cases where $p >> n$, the selection process becomes even harder, and in cases where n is at the order of tens or hundreds of thousands, big data is such an example, the task becomes even more computationally demanding. On the other hand, when n is is small, a quite common phenomenon in datasets occurring in biomedical or clinical research, special treatment, such as re-sampling based testing approaches for example, is required.

There are many reasons why researchers and practitioners should perform feature selection [17]. Many features may be expensive (and unnecessary) to measure, especially in the clinical and medical domains. Parsimonious models are computationally cheaper and easier to understand and interpret. Future experiments will benefit from prior feature selection tasks and give more insight about the problem of interest, its characteristics and structure.

Feature selection can also be used as a means of knowledge discovery and for gaining intuition on the data generation mechanisms. There is theoretical connection between feature selection and the Bayesian (causal) network that describes best the data at hand [17]. Following the Bayesian networks terminology, the Markov Blanket of a variable Y is defined as the minimal set of variables that renders all other variables conditionally independent of Y. That is, the Markov Blanket of Y carries all the necessary information about Y, and no other variable offers additional information about Y. Under certain broad conditions the Markov Blanket has been shown to be the solution to the FS problem [19]. Identification of the Markov Blanket of the target variable is often the primary goal of FS and not the predictive model per se [3]. This is particularly true in medicine,

where the features selected may direct future experiments and offer useful insight on the problem of interest, its characteristics and structure.

MXM [8] is an R package[1] that can be downloaded from CRAN. It contains many feature selection algorithms designed for small and large sample sized datasets. *MXM* offers (Bayesian) network construction as well, but in this paper we focus on its feature selection algorithms. For every algorithm, we discuss their criteria for selecting features (statistical, information based, ad-hoc), computational tricks that can decrease execution time substantially, and share a comprehensive list that summarizes the different types of data these algorithms can be used with. We show how to use some of these algorithms, and some extra useful functions, using real high dimensional biological data.

The rest of the paper is organized as follows. The next Section presents the feature selection algorithms available in *MXM* along with some brief discussion of their properties and the types of data they can handle. Section 3 provides an overview of these algorithms in R, while Section 4 demonstrates some of these algorithms in R. We perform a comparative analysis using gen expression data in Section 4 and Section 5 concludes the paper.

2 Overview of the feature selection algorithms and types of data MXM can handle

MMPC [18] and SES [16] are two examples of algorithms designed mainly for small sample sizes, whereas gOMP [12] and FBED [3] should be preferred for datasets with large sample sizes. Below we give an overview of all the available feature selection algorithms in MXM.

2.1 Forward and backward search

The first, historically, feature selection algorithms are the forward search (FS) and backward search (BS) or backward elimination. FS's first step consists of finding the single predictor variable, mostly associated with the target variable. The association is measured via fitting a regression model for each feature. The procedure continues with the addition of the predictor variables (one at the time) whose contribution, given the selected variables, is significant (again by fitting a regression model for each non selected predictor variable given the selected variables). BS on the other hand works the opposite way. It fits a regression model with all variables and then eliminates, one at the time, the least important variables. Again, many regression models must be fitted at each step.

2.2 IAMB

Incremental Association Markov Blanket (IAMB) [19] is a BN aspired algorithm which consists of two phases; The FS followed by a modified backward elimination. Instead of removing the least significant variable, among the non significant ones, at the time, IAMB removes all of the non significant ones at every step.

[1] *MXM* was first launched in 12/5/2014 and its current version is 1.3.3 (30/3/2018).

2.3 FBED

The most recently suggested feature selection algorithm is Forward-Backward with Early Dropping (FBED) [3] which is also BN inspired. The forward phase step is the same as in the FS with the difference that non (statistically) significant variables are removed and not examined in the subsequent steps (the Early Dropping heuristic). The process continues until no predictor variable can be selected or all of them have been removed. FBED discovers a subset of the MB, parents and children in specific.

Re-running the algorithm again, starting with the selected variables and examining all the rest, results in identifying the MB of the target variable [3], spouses are now identified, while with more re-runs FBED identifies supersets of MB[2] thus providing a broader set of variables of interests, which are also desirable in many research fields. A BS is applied afterwards to remove any falsely selected variables. [3] based upon experimental evaluations showed that FBED produced comparable results to LASSO [14].

2.4 MMPC and SES

Max-Min Parents and Children (MMPC) [18] is a BN inspired algorithm which tries to identify a subset of the Markov Blanket, namely the parents and children only. In low sample size situations, statistical tests have low power which, in conjunction with an increased number of a estimable parameters increase the probability of obtaining wrong results. The basic principle of MMPC is to perform, more, statistical tests conditioning on subsets of the the selected variables, and hence estimate less parameters than the previous algorithms. It also consists of a forward and backward search as well. The fact that it performs more tests does not make it computationally prohibitive, due to several computational efficiency heuristics and tricks that have been employed, described later.

A second advantage of MMPC is its rationale for choosing the next variable to include using the minimum of the maximum p-values. This has been proved to control the False Discovery Rate [20] and also one can estimate it.

Its generalization, called SES (Statistically Equivalent Signatures) [16, 8] builds upon the ideas of MMPC in order to return multiple (statistically equivalent) sets of predictor variables. SES is one of the few feature selection algorithms suggested in the literature which find multiple sets of predictor variables. In essence, this can be found to be of high importance, in cases where some variables are difficult or expensive to be tracked and measured, as for example some biomarkers of interest in terms of clinical research.

MMPC has been successfully applied to survival data [11] and in the context of classification tasks [10]. It has also been applied in integrative feature selection where multiple independent datasets are available [9]. SES was used in high dimensional data with repeated measurements [15] outperforming LASSO algorithm [6] both in predictive performance and computational cost. Groll and Tutz's LASSO implementation [6, 5] has a computational limit in R, but even for a

[2]Re-running the algorithm until no more variables can be included results in the MB of a Maximal Ancestral Graph (MAG) [3], a broader class of BNs.

few hundreds of variables it was shown to be computationally more expensive than SES (and of MMPC essentially).

2.5 MMMB

Max-Min Markov Blanket (MMMB) [18] was proposed as an improvement of MMPC, since it tries to identify the parents of the children discovered by MMPC. It repeatedly performs MMPC on the each of the selected variables of MMPC, in order to identify the spouses of the target variable.

2.6 gOMP

Orthogonal Matching Pursuit [12] is an ad hoc, or geometry based, forward procedure which does not rely on statistical tests. The search begins by selecting the predictor variable mostly correlated with the target variable, i.e. the predictor variable whose angle with the target variable is minimum. A regression model is fitted and the residuals are computed. The correlation between the residuals and the other predictor variables is calculated. The variable with the highest correlation (or smallest angle) is selected. A model is fitted, residuals are computed and this process is repeated. The algorithm stops when the norm of the residuals is below a user-defined threshold, or when some other user-defined criterion is satisfied. OMP is closely related to Forward Stepwise Regression [23] which is based on statistical tests, and also to Forward Stagewise Regression [7].

2 described some differences between OMP and the classical forward selection method. The main difference mentioned in their work, which is in fact the same difference between OMP and FBED, is that OMP selects the new feature based on the inner product, i.e. the angle between the features and the current residual. On the contrary, FBED (and similar model fitting algorithms) will select the feature with the smallest angle, followed by the projection of this feature onto the orthogonal subspace in such a way so as to maximize an objective function, e.g. log-likelihood. 2 made this difference clear using a graphical example.

Our modification of OMP, called generalised OMP (gOMP) uses a adifferent stopping criterion. When a new feature is chosen for possible inclusion, the model with that feature is fitted and the log-likelihood is calculated. If the increase in the log-likelihood values, between the current model and the previous model is above a certain threshold [3], the feature is selected, otherwise the whole process stops. Furthermore, we have implemented gOMP (available in the *MXM* R package) to accept a numerous types of outcome variables, including multi-class, survival, left censored, counts and proportions to name a few.

2.7 Time complexity of the algorithms

We measure the time complexity of the algorithm in terms of the number of performed tests. The number of tests FS performs is $p \cdot (s+1)$ where p is the number of variables and s the number of selected variables. BS on the other hand requires $p + p - 1 + ... + p - s$ tests. IAMB requires $p \cdot (s+1)$ tests for the forward phase while its upper limit for the backward phase is the tests

[3] An example of the threshold is the 95% quantile of the χ^2 distribution, forming a hypothesis test at the 5% significance level.

required by BS. FBED's time complexity is bounded (worst case scenario) by the complexity of FS + BS.

MMPC (and SES) share the same upper bounds for the time complexity. Each variable must be contrasted against each subset of the selected signature before being eliminated. This would require a number of tests in the order of $O\left(p \cdot 2^s\right)$. However, we only allow conditioning upon subsets of the selected variables with maximum cardinality equal to k, at each time, decreasing the complexity of the algorithm to $O\left(p \cdot s^k\right)$. This means that the algorithm can still require an exponential number of tests with respect to the size of the selected signatures; however, in our experience the actual computational requirements of the algorithm are much lower, also due to the parsimonious signatures often retrieved.

Among all algorithms implemented in *MXM*, gOMP is the fastest and computationally cheapest, since unlike all other aforementioned methods, a regression model is fitted only when a variable is selected for possible inclusion. All other methods fit regression models every time a test is performed. In gOMP, only $s + 1$ regression models will be fitted. We must add though that at each step, a vector of correlations is calculated, which however adds a small computational burden. This means, that the number of times parameters are estimated is the same as in FS, yet correlations are easier to compute, than regression models, as they require less computations.

2.8 Types of data *MXM* can handle

In this Section we describe a few types of target variables and their respective regression models suitable for each case.

1. **Continuous** outcome variable (taking any value in the whole of $\mathbb{R}$) are handled with
   - *linear regression*
   - *MM estimators* (linear regression robust to outliers) and
   - *quantile (median) regression* (linear regression robust to outliers).
   - When the predictor variables are also continuous *Pearson* correlation coefficient or
   - *Spearman* correlation coefficient are faster alternatives.
2. **Multivariate continuous outcomes** are handled with
   - *multivariate linear regression.*
   - Compositional data [1] excluding 0 and 1 values are handled the same way since the additive log-ratio transformation is applied.
3. **Positive valued data** are modeled with
   - *Gaussian regression with a log-link*, allowing for zero values or
   - *Gamma regression*, excluding zero values.
4. **Percentages**, are modeled by

- *Beta regression* excluding 0s and 1s.
- Alternatively, one can apply the logit transformation and treat them as continuous and hence use all models mentioned in (1).
- A more general (without making parametric assumptions and accepting 0s and 1s) is the *quasi binomial regression.*

5. **Count data** are usually modeled by
   - *negative binomial regression.*
   - Other options include the more restrictive *Poisson regression* or
   - the more flexible, *quasi Poisson regression.*
   - If an excessive number of zeros has been observed, *zero inflated Poisson regression* should be used instead.
6. **Binary, nominal or of ordinal scale** outcome variables,
   - require *logistic, multinomial and ordinal regression* respectively.
   - If the predictor variables are also categorical, the G^2 test of independence can be used as well, which has the advantage over regression models that "captures" non linear dependencies as well. Its disadvantage is that it requires large sample sizes.
7. **Success out of a number of trials** are modeled by *binomial regression.*
8. **Time to event** (usually right censored, survival) outcomes are handled with survival regression models,
   - *Weibull regression* (parametric model),
   - *exponential regression* (parametric model) or with
   - *Cox regression* (semi-parametric regression model).
   - *Mixed effects Cox regression* (Cox regression for clustered data).
9. **Matched case-control studies** require conditional logistic regression.
10. **Left censored** outcomes are handled by Tobit regression.
11. **Repeated measurements**, over time for example, require parametric models such as
    - *generalized linear mixed models* (GLMM) including
      - *Gaussian regression*,
      - *logistic regression*,
      - *Poisson regression*,
      - *Gamma regression*,
      - *Gaussian regression with log-link* and

- *ordinal regression* but with limited options.

- Distribution free models, such as *generalized estimating equations* (GEE) is a competing to GLMM methodology. The GEE methodology available in *MXM* is richer than GLMM; it includes the same regression models as GLMM and
  - It offers more types of correlation structure (*compound* and *autoregressive*) and many ways of estimating the standard error of the regression coefficients (*standard sandwich estimators*, *jackknife* and *approximate jackknife estimators*).

12. **Classification with longitudinal data**. This is the case when one wants to discriminate among two or more classes and the predictor variables consist of longitudinal data.

3 Overview of the algorithms in R

We will now show which are the relevant commands and discuss their input arguments and output. The types of tests we will briefly mention here are statistical (likelihood ratio test, F test, Wald test, permutations based test) or information criteria (eBIC and BIC).

At this point we should mention that before an algorithm is run, a check for missing data is performed and if there are any they are replaced by the variable's median value (in case of continuous data) or by the variable's mode (in case of categorical data). In addition, variables with constant values (zero variance) are handled internally without damaging the sequence of the variables.

With user-friendliness in mind, extra attention has been put in keeping the functions within the MXM package as consistent as the nature of the algorithms allows for, in terms of syntax, required input objects and parameter arguments.

3.1 MMPC

The function for MMPC is called as:

```
> MMPC(target, dataset, max_k, threshold, test,
ini, wei, user_test, hash, hashObject, ncores,
backward = FALSE)
```

Below we briefly explain some of these arguments. More information on all arguments is available at the *MXM*'s reference manual.

- **test**: The statistical test to use. Default value is NULL. In Section 2.8 we listed the different types of target variables and the respective regression models. The names of the tests here are connected with those regression models. Some example options are "testIndFisher" (Fisher test using the Pearson correlation) "testIndReg" (for linear regression), "testIndPois" (Poisson regression). In general, the test for all algorithms, is termed "testInd..." followed by an acronym or the name of the regression model. The exception is with "censIndCR",

"censIndWR" and "censIndER" which are used for Cox, Weibull and exponential regression respectively.

- **ini**: This is a list with the univariate associations, the test statistic and its p-value. This can help avoid calculating the univariate associations, in case of a second run of MMPC or other algorithms. For example, if you have ran FBED, you can take that list and plug it in here or in MMPC, or the other way round. This will speed up the second run (and subsequent runs), especially in case of demanding regression models, such as GLMM or GEE.

- **user_test**: A user-defined conditional independence test (provide a closure type object). If the user wants to use his own test, he can pass the name of the function here. The function must accept the same input arguments and return the same output as the test functions of *MXM*.

- **hash**: A boolean variable which indicates whether (TRUE) or not (FALSE) to store the statistics calculated during SES execution in a hash-type object.

- **hashObject**: A List with the hash objects generated in a previous run of SES or MMPC. Each time SES runs with "hash=TRUE" it produces a list of hashObjects that can be re-used in order to speed up next runs of SES or MMPC. An important note is that the generated hashObjects should be used only when the same dataset is re-analyzed, possibly with different values of **max_k** and **threshold**. This computational trick can help save a significant amount of time in subsequent runs of MMPC or SES.

The output of MMPC is long verbosed, as it contains the test statistic values and their corresponding p-values, both during execution of the algorithm and of the univariate associations. The latter is stored in a list called *univ* and can be passed on to other algorithms such as FBED. Other returned items include the *selected variables*, the *statistically equivalent signatures* (if any), the **hashObject** and the *number of tests performed* (if **hash** was set to TRUE), the natural logarithm *(maximum) p-value* of each performed test, the relevant *test statistic value*, the *execution time* and the **test** used.

MMPC (and SES) make use of statistical tests in order to include a variable. These tests are either the F-test or the likelihood ratio test depending on the regression model. We have also implemented MMPC and SES with the Wald test, currently designed for continuous predictor variables only. The functions are called **waldMMPC** and **waldSES** and the names of the tests to be passed in argument **test** are similar to all other functions, but begin with *wald*: **waldMMReg**, **waldPois**, **waldLogistic** etc.

A third alternative, is to use permutation based statistical tests. In these tests, the p-value is evaluated using permutations (default value is 999). In order to decrease the computational cost, we stop the the permutations once the p-value exceeds the threshold value. For example, if the threshold value is set to 0.05 and the number of permutations is equal to 999, if the p-value after 51 permutations has exceeded 0.05 we do not perform the rest of the permutations. The names of the tests in this case are **permFisher**, **permReg**, **permLogistic**, etc. and are passed to **permMMPC** or **permSES**.

Generalized linear mixed models (GLMM) and GEE (for time course, longitudinal or clustered data) have their own functions, called **SES.temporal** and **SES.gee** respectively. The signature is the same in all cases (GLMM, GEE, Wald based or permutation based MMPC and SES) with the addition of some extra arguments. For the GEE here is an example of the extra arguments.

```
> MMPC.gee(target, ..., correl = "exchangeable",
se = "jack", ncores)
```

The user has the option to choose the correlation structure (*correl*) and the type of estimation of the standard error of the regression coefficients (*se*). The names for GLMM and GEE are similar to MMPC. For example **testIndGLMMReg**, **testIndGLMMPois** and **testIndGEEReg**, **testIndGEEPois**.

3.2 FBED

The input arguments for FBED are different because of the nature of the algorithm.

```
> fbed.reg(target, dataset, test, threshold, wei, K,
method, gam, backward)
```

Let us now briefly explain some of the input arguments.

- **ini**: If you already have the test statistics and the p-values of the univariate associations (the first step of FBED) supply them as a list with the names *stat* and *pvalue* respectively. This is the same as the **ini** in SES. As mentioned before, this can be extracted from a SES or MMPC run, hence reducing the computational cost. In the case of FBED having been called with eBIC as the criterion of inclusion (**method**=eBIC), then this is a list with one element, called *ebic* and can only be used by FBED.
- **K**: How many times should the process be repeated? The default value is 0. The user has the option to pass a range of values of K, such as *0:4* in which case the algorithm will return the selected variables for each value of K.
- **method**: This argument give the user the choice of the inclusion test. In case of statistical test (F-test or likelihood ratio test, whichever appropriate) the user should type "LR". For information criteria, such as "eBIC" (extended BIC, [4]) or the usual BIC, the user should type "eBIC".
- **gam**: In case the method is chosen to be "eBIC" one can also specify the γ parameter of the eBIC. The default value is "NULL", so that the value is automatically calculated. If the user sets this argument equal to 0, the usual BIC is used.

In the case of GEE the command is called as

```
> fbed.gee.reg(target, dataset, id, ...,
correl = "exchangeable", se = "jack")
```

The output of FBED depends upon whether the user has run FBED with a single value of K or not. In some, the univariate associations tests, the selected variables along with the value of the selected criterion (statistical test or information criterion) and the running time are returned. Information about the number of selected variables and number of tests performed for each value of K and in the case of more than one value of K, a list with information on each value of K is also returned.

3.3 gOMP

The command for gOMP is different than all the other commands, due to the nature of the algorithm.

```
> gomp(target, dataset, tol, test)
```

All arguments are the same as before, except for **tol**. This is the criterion to include a variable and is based upon the log-likelihood of the fitted regression model at each time. If twice the difference between two successive log-likelihoods is less than the tolerance value the algorithm stops. A tolerance value equal to 3.84 corresponds to0 the log-likelihood ratio test, whereas a value of $3.84 + log(n)$, where n is the sample size, corresponds to using BIC as the test of inclusion. There is no *ini* argument here due to the algorithm being very fast.

gOMP's output is very small and it includes the running time of the algorithm, the selected variables and the deviance or twice the log-likelihood of the model at each step.

4 Demonstration of the algorithms using real data

We will demonstrate MMPC, FBED and gOMP using 2 real datasets [4]. The first dataset concerns breast cancer [22] and contains 285 samples over $17,187$ features. The target variable is binary, hence logistic regression (**test = "testIndLogistic"**) will be employed. The second one regards survival times (again breast cancer related) [21] and contains 78 observations, out of which 44 are censored, with $4,571$ features. Cox regression will be used (**test = "censIndCR"**). All computations took place in a desktop computer with Intel Core i5-4690K CPU @3.50GHz and 32 GB RAM.

4.1 Breast cancer dataset

4.1.1 MMPC

[4] The case of SES, was covered in [8].

```
> mmpc1 <- MMPC(target, dataset, max_k = 3, hash = TRUE,
test = "testIndLogistic")
```

We then re-run MMPC with $max\_k = 4$, but taking into consideration the previous calculations.

```
> mmpc2 <- MMPC(target, dataset, max_k = 4, hash = TRUE,
ini = mmpc1@univ, hashObject = mmpc1@hashObject,
test = "testIndLogistic")
```

The output is very long verbosed and hence omitted. The running time required the first time (**mmpc1**) is equal to 70 seconds, while performing $31,385$ likelihood ratio tests. In the second run, **mmpc2** required only 8 seconds. If the second run was performed independently from the previous run, (performing all tests again) the running time would be equal to 75 seconds, while performing $32,021$ statistical tests. It is then obvious that **mmpc2** performed only 636 extra tests. There is a non negligible time required to search and extract the information about a test from a hash object, yet, this time is not comparable to the time the performance of a statistical test requires.

Since **hash** was set to TRUE, by using the command **certificate.of.exclusion** the user can detect the conditioning variables (if any) responsible for not including a specific predictor variable. In this example, the user might be interested in seeing why a specific gene was not selected. The first variable for example, even though has a statistically significant unconditional association with the target variable, it was not selected by MMPC.

```
> certificate.of.exclusion(1, mmpcObject = mmpc1)
$`1`
         <NA>     statistic       p-value
4509.00000000    0.09795647   -0.28197058
```

The association between the outcome variable and the first feature becomes non significant, when conditioning on the 4509th variable and its p-value is $exp(-0.28197058) = 0.7542959$.

Throughout all the algorithms the logarithm of the p-values is calculated and stored. When the p-value is less than 10^{-16} (machine epsilon), R rounds the p-value to 0. When there are more than one p-value rounded to 0, the smallest p-value is decided at random. If this situation takes place very often inside the algorithm the final output might no be what is expected. Since, all p-value based algorithms are based on ordering the p-values, this is a crucial and key element of the algorithms and caution, or the logarithm of the p-value, must be taken.

Suppose now one is interested in running MMPC for many combinations of **max_k** and **threshold**. This can be done by using **mmpc.path** which is essentially a hyper-parameter tuning, grid search function, for choosing the optimal configuration of the the **threshold** and **max_k**. The

output for this dataset is

```
$bic
            max_k=4  max_k=3  max_k=2
alpha=0.05 111.6808 125.5517 146.9647
alpha=0.01 116.4926 116.4926 126.2583

$size
           max_k=4 max_k=3 max_k=2
alpha=0.05       8      13      25
alpha=0.01       6       6      10
```

The element *bic* contains the BIC for the regression model applied to the selected features using each combination of **max_k** and **threshold**. The element *size* shows the number of the selected features. The extra element (not shown here) is another list with the selected features for each combination of **max_k** and **threshold**. As for the running time, that was only 160 seconds. This function takes into account the computational benefits of the hash tables and for every subsequent runs of MMPC performs fewer tests.

4.1.2 FBED

FBED using likelihood ratio tests and eBIC were run as well with K=0. This means that no repetitions were performed.

```
> fbed1 <- fbed.reg(target, dataset,
method = "LR", test = "tesTIndLogistic")

fbed1$res
     sel       stat       pval
1   4509 101.490982 -53.290889
2  17606  52.394553 -28.420720
3   3856  26.829649 -15.319679
4  16361  12.597669  -7.859082
5  11797  11.643982  -7.347647
6  16837  10.053095  -6.488439
7   3157  18.446479 -10.954777
8   3400   6.693620  -4.638122
9   1457   9.695889  -6.294297
10  4404   5.267671  -3.829289

> fbed2 <- fbed.reg(target, dataset,
```

```
test = "tesTIndLogistic", method = "eBIC")

fbed2$res
      Vars     eBIC
[1,]  4509 247.0451
[2,] 17606 149.3605
[3,]  3856 127.9701
[4,] 16361 123.8473
[5,] 11797 128.0821
[6,]  4404 120.8556
```

One element of the returned output is displayed above. The p-value based selection (likelihood ratio test) required 55 seconds, performed 28,688 tests and selected 10 variables. The eBIC based selection required 45 seconds, performed 23,641 tests and selected 6 variables.

4.1.3 gOMP

gOMP required 1.5 seconds and selected 9 variables.

```
> gomp(target, dataset, test = "testIndLogistic")
```

The element *res* presented below is one of the elements of the returned output. The first column shows the selected variables in order of inclusion and the second column is the deviance of each regression model. The first line refers to the regression model with 0 predictor variables (constant term only).

```
$res
      Selected Vars  Deviance
 [1,]             0 332.55696
 [2,]          4509 156.33519
 [3,]         17606 131.04428
 [4,]          3856 113.78382
 [5,]         10101  95.76704
 [6,]         16759  80.25748
 [7,]          6466  67.78120
 [8,]         11524  54.54652
 [9,]          9794  44.17957
[10,]          4728  36.52319
```

4.2 Survival dataset

The target variable must be transformed into a **surv** object using the command **Surv** in the package *survival* [13].

4.2.1 MMPC

MMPC selected the following features

```
> mmpc <- MMPC(target, dataset, test = "censIndCR")
ses@selectedVars
[1,] 1458 3872 3945 4554
```

A useful command is **mmpc.model** which returns the regression model with the features selected by MMPC. The output for this dataset is

```
> mmpc.model(target, dataset, mmpcObject = mmpc)
$mod
Call:
survival::coxph(formula = target ~ ., data = data.frame(dataset[,
    signature]), weights = wei)

          coef exp(coef) se(coef)     z       p
V1460   3.9013   49.4644   1.0176  3.83 0.00013
V3874  -4.5911    0.0101   1.6634 -2.76 0.00578
V3947  -3.5580    0.0285   0.8700 -4.09 4.3e-05
V4556   5.6801  292.9789   1.7448  3.26 0.00113

Likelihood ratio test=46.6  on 4 df, p=1.82e-09
n= 78, number of events= 44

$signature
V1460 V3874 V3947 V4556   bic
 1458  3872  3945  4554   219
```

4.2.2 FBED

The selected variables discovered by FBED using likelihood ratio test and eBIC (they are the same in this dataset)

```
fbed1$res
   sel      stat       pval
```

```
1 3872 12.027483  -6.013742
2 3945 22.729195 -11.364597
3 1458 13.540375  -6.770188
4 4554 12.912292  -6.456146
5 4325  8.072538  -4.036269
6  386  7.128295  -3.564147
```

and

```
fbed2$res
     Vars     eBIC
[1,] 3872 222.0323
[2,] 3945 232.7340
[3,] 1458 223.5452
[4,] 4554 222.9171
[5,] 4325 218.0773
[6,]  386 217.1331
```

respectively. We can see that both methods selected the same variables.

Similarly to the command **mmpc.path** applied to MMPC, FBED gives the user the option to output all results for different values of K. As mentioned in the description of FBED, the argument **K** determines the number of re-runs of FBED. The default value is 0. The user can either specify a single value for K, or a range of values of K and gather the path of solutions for many values of re-runs. For example, by calling **fbed.reg** with **K=0:3** we get the following results

```
$res$info
    Number of vars Number of tests
K=0              6             5073
K=1             10             4885
K=2             16             5198
K=3             21             4988

$mod$`K=0`
  Vars      stat log p-value
1 3872 12.027483   -6.013742
2 3945 22.729195  -11.364597
3 1458 13.540375   -6.770188
4 4554 12.912292   -6.456146
5 4325  8.072538   -4.036269
```

```
6  386  7.128295   -3.564147

$mod$`K=1`
   Vars      stat log p-value
1  3872 37.157121  -18.578561
2  3945 16.746585   -8.373293
3  1458 38.083482  -19.041741
4  4554 11.794991   -5.897496
5  4325 11.519512   -5.759756
6   386 27.296062  -13.648031
7  3805 15.395575   -7.697788
8  4689 16.185231   -8.092616
9  1682 12.446433   -6.223216
10 2355  6.544932   -3.272466

$mod$`K=2` (committed)
$mod$`K=3` (ommitted)
```

We re-ran FBED using *K=1* and called the command **fbedreg.bic** which takes as inputs the target variable, the dataset, the object where the FBED results are saved, the test used (**"censIndCR"** in our case). The function plots the BIC a function of the selected variables in the order they were selected. The produced plot is presented in Figure 1. The results of the FBED run along with the relevant BIC values and the final regression model are two arguments (not presented here) returned as well.

4.2.3 gOMP

As for gOMP the results are

```
      Selected Vars Deviance
 [1,]             0 250.6345
 [2,]          3945 240.3924
 [3,]          2874 225.6243
 [4,]           520 209.1400
 [5,]          4402 200.6827
 [6,]          1467 186.0771
 [7,]          3212 173.5159
 [8,]          4130 151.6564
 [9,]          3095 146.3925
```

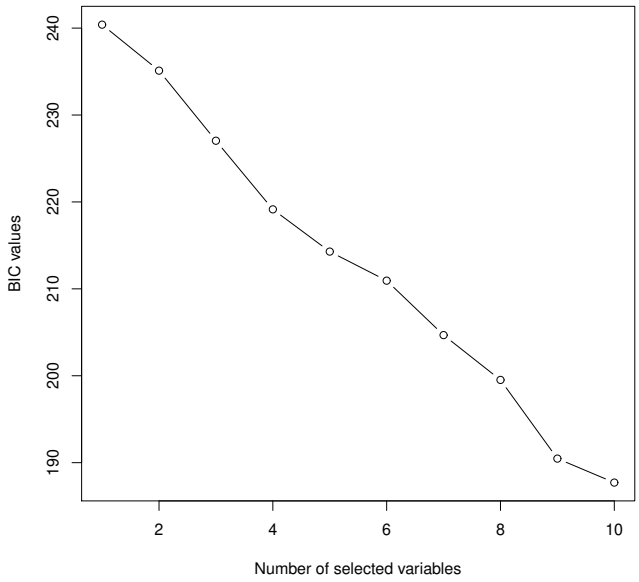

Figure 1: Successive BIC values for the Cox regression using the selected variables in order of inclusion.

A final command not mentioned earlier is the **reg.fit** which accepts any type of target variable and fits any regression model currently available in *MXM*. This is more general than **mmpc.model** and **ses.model** since it does not depend on the output of any algorithm. The command **reg.fit** has a long list of arguments, covering many regression algorithms. Cross-validation procedures currently exist for *MMPC* and *SES* only (and their *Wald* and *perm* versions). The relevant commands are **cv.mmpc** and **cv.ses** respectively.

Acknowledgments

The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement n. 617393.

References

[1] John Aitchison. *The statistical analysis of compositional data*. New Jersey: Reprinted by The Blackburn Press, 2003.

[2] Thomas Blumensath and Mike E Davies. On the difference between orthogonal matching pursuit and orthogonal least squares. Technical report, University of Southampton, UK, 2007.

[3] Giorgos Borboudakis and Ioannis Tsamardinos. Forward-backward selection with early dropping. *arXiv preprint arXiv:1705.10770*, 2017.

[4] Jiahua Chen and Zehua Chen. Extended bayesian information criteria for model selection with large model spaces. *Biometrika*, 95(3):759–771, 2008.

[5] Andreas Groll. *glmmLasso: Variable Selection for Generalized Linear Mixed Models by L1-Penalized Estimation*, 2015. R package version 1.3.4.

[6] Andreas Groll and Gerhard Tutz. Variable selection for generalized linear mixed models by L_1-penalized estimation. *Statistics and Computing*, 24(2):137–154, 2014.

[7] Trevor Hastie, Jonathan Taylor, Robert Tibshirani, Guenther Walther, et al. Forward stagewise regression and the monotone lasso. *Electronic Journal of Statistics*, 1:1–29, 2007.

[8] Vincenzo Lagani, Giorgos Athineou, Alessio Farcomeni, Michail Tsagris, and Ioannis Tsamardinos. Feature Selection with the R Package MXM: Discovering Statistically-Equivalent Feature Subsets. *Journal of Statistical Software*, 80, 2017.

[9] Vincenzo Lagani, Argyro D Karozou, David Gomez-Cabrero, Gilad Silberberg, and Ioannis Tsamardinos. A comparative evaluation of data-merging and meta-analysis methods for reconstructing gene-gene interactions. *BMC bioinformatics*, 17(5):S194, 2016.

[10] Vincenzo Lagani, George Kortas, and Ioannis Tsamardinos. Biomarker signature identification in omics data with multi-class outcome. *Computational and structural biotechnology journal*, 6(7):1–7, 2013.

[11] Vincenzo Lagani and Ioannis Tsamardinos. Structure-based variable selection for survival data. *Bioinformatics*, 26(15):1887–1894, 2010.

[12] Yagyensh Chandra Pati, Ramin Rezaiifar, and Perinkulam Sambamurthy Krishnaprasad. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *Signals, Systems and Computers, 1993. 1993 Conference Record of The Twenty-Seventh Asilomar Conference on*, pages 40–44. IEEE, 1993.

[13] Terry Therneau and Thomas Lumley. Survival: Survival analysis, including penalised likelihood. r package version 2.36-5. *Survival: Survival analysis, including penalised likelihood. R package version*, pages 2–36, 2011.

[14] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996.

[15] Michail Tsagris, Vincenzo Lagani, and Ioannis Tsamardinos. Feature selection for high-dimensional temporal data. *BMC bioinformatics*, 19(1):17, 2018.

[16] I. Tsamardinos, V. Lagani, and D. Pappas. Discovering multiple, equivalent biomarker signatures. In *In Proceedings of the 7th conference of the Hellenic Society for Computational Biology & Bioinformatics, Heraklion, Crete, Greece*, 2012.

[17] Ioannis Tsamardinos and Constantin F Aliferis. Towards principled feature selection: relevancy, filters and wrappers. In *AISTATS*, 2003.

[18] Ioannis Tsamardinos, Constantin F Aliferis, and Alexander Statnikov. Time and sample efficient discovery of Markov Blankets and direct causal relations. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 673–678. ACM, 2003.

[19] Ioannis Tsamardinos, Constantin F Aliferis, Alexander R Statnikov, and Er Statnikov. Algorithms for Large Scale Markov Blanket Discovery. In *FLAIRS Conference*, volume 2, 2003.

[20] Ioannis Tsamardinos and Laura E Brown. Bounding the False Discovery Rate in Local Bayesian Network Learning. In *AAAI*, pages 1100–1105, 2008.

[21] Laura J Van't Veer, Hongyue Dai, Marc J Van De Vijver, Yudong D He, Augustinus AM Hart, Mao Mao, Hans L Peterse, Karin Van Der Kooy, Matthew J Marton, Anke T Witteveen, et al. Gene expression profiling predicts clinical outcome of breast cancer. *nature*, 415(6871):530, 2002.

[22] Yixin Wang, Jan GM Klijn, Yi Zhang, Anieta M Sieuwerts, Maxime P Look, Fei Yang, Dmitri Talantov, Mieke Timmermans, Marion E Meijer-van Gelder, Jack Yu, et al. Gene-expression profiles to predict distant metastasis of lymph-node-negative primary breast cancer. *The Lancet*, 365(9460):671–679, 2005.

[23] Sanford Weisberg. *Applied linear regression*. John Wiley & Sons, New York, 1980.