

# Selection of Significant Features Using Monte Carlo Feature Selection

Susanne Bornelöv and Jan Komorowski

**Abstract** Feature selection methods identify subsets of features in large datasets. Such methods have become popular in data-intensive areas, and performing feature selection prior to model construction may reduce the computational cost and improve the model quality. Monte Carlo Feature Selection (MCFS) is a feature selection method aimed at finding features to use for classification. Here we suggest a strategy using a z-test to compute the significance of a feature using MCFS. We have used simulated data with both informative and random features, and compared the z-test with a permutation test and a test implemented into the MCFS software. The z-test had a higher agreement with the permutation test compared with the built-in test. Furthermore, it avoided a bias related to the distribution of feature values that may have affected the built-in test. In conclusion, the suggested method has the potential to improve feature selection using MCFS.

**Keywords** Feature selection · MCFS · Monte Carlo · Feature significance · Classification

## 1 Introduction

With the growth of large datasets in areas such as bioinformatics, computational chemistry, and text recognition, limitations in the computational resources may force us to restrict the analysis to a subset of the data. Feature selection methods reduce the

---

S. Bornelöv · J. Komorowski (✉)  
Department of Cell and Molecular Biology, Science for Life Laboratory,  
Uppsala University, Uppsala, Sweden  
e-mail: jan.komorowski@icm.uu.se

S. Bornelöv  
Department of Medical Biochemistry and Microbiology, Uppsala University, Uppsala, Sweden  
e-mail: susanne.bornelov@imbim.uu.se

J. Komorowski  
Institute of Computer Science, Polish Academy of Sciences, Warsaw, Poland

data by selecting a subset of the features. An assumption in feature selection is that large datasets contain some redundant or non-informative features. If successfully removing those, both the speed of the model training, the performance, and the interpretation of the model may be improved [1].

There are several feature selection methods available. For a review of feature selection techniques used in bioinformatics, see Saeys et al. [2]. Some methods are univariate and consider one feature at a time; others include feature interactions to various degrees. In this paper we have studied Monte Carlo Feature Selection (MCFS) [3]. MCFS focuses on selecting features to be used for classification. The use of MCFS was originally illustrated by selecting genes with importance for leukemia and lymphoma [3], and it was later used to study e.g. HIV-1 by selecting residues in the amino acid sequence of reverse transcriptase with importance for drug resistance [4, 5]. Furthermore, MCFS may be used to rank the features based on their relative importance score. Thus, MCFS may be applied even on smaller datasets if the aim is to rank the features by their impact on the outcome (see e.g. [6–8]).

MCFS is a multivariate feature selection method based on random sampling of the original features. Each sample is used to construct a number of decision trees. Each feature is then given a score—relative importance (RI)—according to how it performs in the decision trees. Thus, the selection of a feature is explicitly based on how the feature contributes to classification.

One question is how to efficiently interpret the RI of a feature. If MCFS is used to select a subset suitable for classification, a strategy may be to select the  $x$  highest ranked features [6]. However, a stronger statistical basis for making the cutoff would be preferred, particularly, when MCFS is used to determine which features significantly influence the outcome.

The MCFS algorithm is implemented in the dmLab software available at [9]. There is a statistical test on the significance of a feature implemented in the software. The strategy of the test is to perform a number of permutations of the decision column, and in each permutation save the highest RI observed for any feature. Thereafter, the test compares the RI of each feature in the original data to the 95 % confidence interval of the mean of the best RI scores [5].

Here, we suggest a different methodology that tests each feature separately to its own set of controls. We show that this methodology leads to more accurate results and allows us to identify the most significant feature even when they do not have the highest RI. Furthermore, by testing each feature separately, we avoid biases related to the distribution of feature values. Our suggested methodology is supported by experiments using simulated data.

In conclusion, we have provided a methodology for computing the significance of a feature using MCFS. We have shown that this methodology improves the currently used statistical test, and discussed the implications of using alternative methods.

## 2 Materials and Methods

### 2.1 Monte Carlo Feature Selection

The MCFS algorithm is based on extensive use of decision trees. The general idea is to select  $s$  subsets of the original  $d$  features, each with a random selection of  $m$  features. Each such subset is divided into a training and test set with 2/3 and 1/3 of the objects, respectively. This division is repeated  $t$  times, and a decision tree classifier is trained on each training set. In all,  $st$  decision trees are trained and evaluated on their respective test set. An overview of the methodology is shown in Fig. 1.

Each feature is scored according to how it performs in these classifiers by a score called relative importance (RI). The RI of a feature  $g$  was defined by Draminski et al. [3] as

$$RI_g = \frac{1}{M_g} \sum_{\tau=1}^{st} (wAcc_{\tau})^u \sum_{n_g(\tau)} IG(n_g(\tau)) \left( \frac{\text{no.in } n_g(\tau)}{\text{no.in } \tau} \right)^v \quad (1)$$

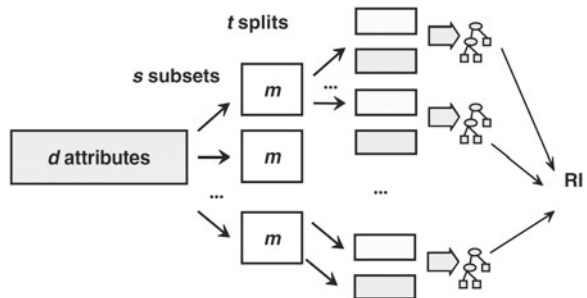
where  $s$  is the number of subsets and  $t$  is the number of splits for each subset.  $M_g$  is the number of times the attribute  $g$  was present in the training set used to construct a decision tree. For each tree  $\tau$  the weighted accuracy  $wAcc$  is calculated as the mean sensitivity over all decision classes, using

$$wAcc = \frac{1}{c} \sum_{i=1}^c \frac{n_{ii}}{n_{i1} + n_{i2} + \dots + n_{ic}} \quad (2)$$

where  $c$  is the number of decision classes and  $n_{ij}$  is the number of objects from class  $i$  that were classified to class  $j$ .

Furthermore, for each  $n_g(\tau)$  (a node  $n$  in decision tree  $\tau$  that uses attribute  $g$ ) the information gain (IG) of  $n_g(\tau)$  and the fraction of the number of training set objects in (no.in)  $n_g(\tau)$  compared to the number of objects in the tree root is computed. There are two weighting factors  $u$  and  $v$  that determine the importance of the  $wAcc$  and the number of objects in the node.

**Fig. 1** Overview of the MCFS procedure.  
Reproduced from Draminski et al. [3]



## 2.2 Construction of Datasets

To apply MCFS and to compute the significance of the features, we constructed datasets with 120 numerical and 120 binary features. For each type of features, 20 were correlated to the decision and 100 were uncorrelated. The decision class was defined to be binary (0 or 1) with equal frequency of both decisions. The number of simulated objects was set to either 100 or 1,000. Thus, for each object the decision class value was randomly drawn from the discrete uniform distribution  $[0,1]$  prior to generating the attribute values. Detailed description of the attributes is provided in the following sections. To verify that the features with an expected correlation to the decision indeed were correlated, the Pearson correlation between each non-random feature and the decision was computed after the data generation (Table 1).

**Numerical Uncorrelated Features:  $RandNum_0$  to  $RandNum_{99}$ .** The values of a numerical uncorrelated feature ( $RandNum_i$ ,  $0 \leq i \leq 99$ ) were randomly drawn from the discrete uniform distribution  $[1, i + 1]$ . Thus, the indices defined the range of

**Table 1** Pearson correlation between each correlated feature and the decision. Presented for both datasets (100 objects and 1,000 objects) separately

$i$	100 objects		1,000 objects	
	$Num_i$	$Bin_i$	$Num_i$	$Bin_i$
0	0.74	0.96	0.74	0.95
1	0.72	0.94	0.65	0.91
2	0.58	0.86	0.63	0.87
3	0.66	0.84	0.50	0.81
4	0.50	0.77	0.50	0.78
5	0.53	0.73	0.47	0.69
6	0.19	0.60	0.43	0.66
7	0.39	0.64	0.41	0.64
8	0.34	0.56	0.35	0.60
9	0.28	0.54	0.35	0.55
10	0.38	0.39	0.28	0.46
11	0.22	0.41	0.29	0.41
12	0.18	0.33	0.23	0.45
13	0.21	0.30	0.20	0.31
14	0.29	0.33	0.14	0.32
15	0.18	0.19	0.16	0.32
16	0.15	0.31	0.16	0.18
17	-0.01	0.01	0.07	0.14
18	0.08	0.07	0.07	0.15
19	-0.06	-0.02	-0.03	0.05

possible values, which allowed us to test whether the number of possible values for a feature influenced its ranking.

**Numerical Correlated Features:  $Num_0$  to  $Num_{19}$ .** The values of a numerical correlated feature ( $Num_i$ ,  $0 \leq i \leq 19$ ) were defined using the following algorithm: Let  $X$  be a random variable from the continuous uniform distribution  $(0,1)$ . If  $X > (i+1)/21$  the value was selected randomly from the binomial distribution  $B(6, 0.5)$  if  $Decision=0$ , and from  $B(6, 0.5) + 3$  if  $Decision=1$ . Otherwise, if  $X \leq (i+1)/21$ , the value was selected randomly from the uniform distribution  $[0, 9]$ . Thus, low values were indicative of  $Decision=0$  and high values of  $Decision=1$ , with a noise level indicated by the feature index.

**Binary Uncorrelated Features:  $RandBin_0$  to  $RandBin_{99}$ .** The values of a binary uncorrelated feature ( $RandBin_i$ ,  $0 \leq i \leq 99$ ) were defined using the following algorithm: Let  $X$  be a random variable from the continuous uniform distribution  $(0,1)$ . If  $X > (i+1)/101$  the value is 1, otherwise it is 0.

Thus, features with low indices will have ones in excess, features with middle indices will have more even distribution of ones and zeroes, and those with high indices will have zeroes in excess.

**Binary Correlated Features:  $Bin_0$  to  $Bin_{19}$ .** The values of a binary correlated feature ( $Bin_i$ ,  $0 \leq i \leq 19$ ) were defined using the following algorithm: Let  $X_1$  be a random variable from the continuous uniform distribution  $(0,1)$ . If  $X_1 > (i+1)/21$ , the value is equal to the decision. Otherwise it is assigned by drawing another random variable  $X_2$  from the continuous uniform distribution  $(0,1)$ . If  $X_2 > (i+1)/21$ , the value is 1, otherwise it is 0.

### 2.3 Performing the Experiments

The experiments were performed using the dmLab software version 1.85. We applied the rule-of-thumb to set the number of features selected in each subset to  $\sqrt{d}$ , where  $d$  is the total number of features. Thus using 240 features, we used  $m = \sqrt{240} \approx 15$ . The number of subsets was set to  $s = 3,000$  for the permutation runs and  $s = 100,000$  for the original data. The number of trees trained in each subset was set to  $t = 5$  and the number of permutation test runs was set to  $cutPointRuns = 10,000$ . The weighting parameters were set to  $u = 0$  and  $v = 1$ .

There were two main arguments for using a higher number of subsets on the original data. Firstly, ranking of the features in the original data is the most crucial part of the experiment. Therefore, it is generally motivated to focus more of the computational resources onto this step. Secondly, both the z-test and the built-in test require the rankings of the original data to be stable, which is obtained by constructing a high number of subsets.

Setting  $u = 0$  will omit the decision tree accuracy from the calculation of RIs. Indeed, using model performance as a selection criteria may be counter-productive

[10], and our experience is that the inclusion of the accuracy in the calculation of the RI overestimates the importance of all features in the original data compared to the permuted ones. This effect is expected, since the accuracy on the original data will reflect the most predictive features, whereas on the permuted data it will only reflect random variation of the decision trees.

## 2.4 Selection of Significant Features

In this section we present different strategies to estimate the  $p$ -value of the RI of a feature using a permutation test, either alone or in combination with additional tests. Using a traditional permutation test requires thousands of permutations to yield efficient estimates of small  $p$ -values. Thus, alternative tests performing a smaller number of permutations and using these to estimate the underlying distribution may save computational time. The test that is built-in into dmLab employs this strategy and performs a t-test comparing the best RIs obtained during the permutation runs to the RI of a feature on the original data. Here we suggest another approach using a z-test to compute the  $p$ -value by estimating a normal distribution for each feature separately.

During the permutation test the number of permutations,  $N$ , was set to 10,000 to obtain sufficient resolution of the  $p$ -values. The permutation test  $p$ -values were then used as a gold standard to evaluate the build-in test and the suggested z-test. For these tests a substantially smaller number of permutations are needed. Consequently, we used only the 100 first permutation runs to estimate the  $p$ -values using the built-in and the z-test.

**Using a Permutation Test to Select Significant Features.** A permutation test may be applied to compute an approximation of the empirical  $p$ -value of a RI. The null hypothesis is that the RI calculated on the real data is no better than the RIs computed for the permuted data. The empirical  $p$ -value approximates the probability of observing a test statistics at least as extreme as the observed value, assuming that the null hypothesis is true. Typically, a significance level, such as 0.05, is defined and attributes associated with  $p$ -values below this level are considered significantly informative.

Theoretically, the true permutation test  $p$ -value of  $RI = x$  that was measured for a feature  $g$  would be

$$p_{true}(RI_g = x) = \frac{\sum_{i=1}^{N_{all}} \mathbf{I}(RI_g^i \geq x)}{N_{all}} \quad (3)$$

where  $\mathbf{I}$  is the indicator function taking value 1 if the condition is met, and 0 otherwise.  $RI_g^i$  is the RI of the attribute  $g$  in permutation  $i$  and  $N_{all}$  denotes the total number of possible permutations. However, since  $N_{all}$  may be extremely large, only a limited

number of permutations are commonly performed. Furthermore, pseudo-counts are added to avoid  $p$ -values of zero, which are theoretically impossible since at least one possible permutation has to be identical to the original data. Thus, an approximation of the permutation test  $p$ -value is commonly applied, which is based on the  $N$  number of permutations with  $N \ll N_{all}$  using the following expression

$$p(RI_g = x) = \frac{1 + \sum_{i=1}^N \mathbf{I}(RI_g^i \geq x)}{N + 1} \quad (4)$$

**Using a z-Test to Select Significant Features.** By performing  $N$  permutations, each feature receives  $N$  estimates of its relative importance on non-informative data. If  $N > 30$  and the RIs are normally distributed, the distribution mean  $\mu_g$  and standard deviation  $\sigma_g$  of a feature  $g$  may be estimated from the data as

$$\mu_g = \frac{1}{N} \sum_{i=1}^N RI_g^i \quad (5)$$

and

$$\sigma_g = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (RI_g^i - \mu_g)^2} \quad (6)$$

where  $RI_g^i$  is the RI of attribute  $g$  in permutation  $i$ .

Thus, the z-score of the RI for a feature  $g$  on the original data,  $RI_g = x$ , may be computed as

$$z = (x - \mu_g) / \sigma_g. \quad (7)$$

A z-test can be applied to calculate the  $p$ -value associated to a particular z-score. Since no feature is expected to perform significantly worse on the original data compared with the permuted one, an upper-tail  $p$ -value was computed.

**Using the Built-in Test to Select Significant Features.** To compare our results, we also used the combined permutation test implemented in the dmLab software. This test is also based on  $N$  permutations of the decision, and using each such permuted dataset, the whole MCFS procedure is repeated and the RI of each feature is computed. As opposed to the previous strategies, only the highest RI from each permuted dataset ( $RI_{\max}$ ) is used, independently of which feature it is based on. Thus,  $N$  such  $RI_{\max}$  values are generated and used to estimate the parameters  $\mu_{\max}$  and  $\sigma_{\max}$  applying

$$\mu_{\max} = \frac{1}{N} \sum_{i=1}^N RI_{\max}^i \quad (8)$$

and

$$\sigma_{\max} = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (RI_{\max}^i - \mu_{\max})^2}. \quad (9)$$

A  $t$ -statistic is then computed per feature  $g$  as

$$T = (x_g - \mu_{\max}) / (\sigma_{\max} / \sqrt{N}) \quad (10)$$

and the two-sided  $p$ -value associated to the  $t$ -statistics is obtained.

### 3 Results

#### 3.1 Results of Simulation Study

We applied MCFS to the datasets with 100 and 1,000 objects. Table 2 summarizes the results after MCFS using 100 objects. The RI of each feature is reported, as well as the estimated RI mean and standard deviation on the permuted data. The 10,000 RIs computed for each feature on the permuted data were approximately bell shaped, occasionally displaying a bias towards either of the distribution tails. The  $p$ -values were calculated using the z-test and the permutation test as described in Sect. 2.4. Additionally, an overall RI threshold at the 0.05 significance level was estimated to 0.0787 using the built-in method in dmLab.

Using both the z-test and the permutation test  $Num_0$ - $Num_5$ ,  $Num_7$ - $Num_8$ ,  $Num_{10}$ ,  $Bin_0$ - $Bin_{11}$ ,  $Bin_{14}$ , and  $Bin_{16}$  were significant at the 0.05 level. Using the built-in t-test combining all features, the  $Bin_{10}$ - $Bin_{11}$ ,  $Bin_{14}$ , and  $Bin_{16}$  were not identified as significant since their RI was below 0.0787. Note that  $Bin_{16}$  was significant according to the z-test and the permutation test, although it had a lower RI than  $Num_9$  that was not significant using any of the tests.

A notable association between the ranking of the random binary features and their indices was observed (Fig. 2a), where features with intermediate indices were ranked higher than those with low or high indices. Since the random binary features with low or high indices were defined to have an excess of ones or zeroes, respectively, this corresponds to a weak preference for features with a uniform distribution of values. However, no relation between the value range of a feature and its relative importance was observed, consistent with previously reported results [11], although the variation of the RIs increased slightly with the value range (Fig. 2b). Both the binary and numeric features were scored according to their expected relevance (Fig. 2c, d).

Since the data was randomly generated simulating only 100 objects, the exact size of the effect for a feature may differ slightly from the expectation. Thus, we repeated the same methodology with a sample size of 1,000 objects instead. The results are shown in Table 3. This time,  $Bin_0$ - $Bin_{15}$  and  $Num_0$ - $Num_{13}$  were selected using z-test

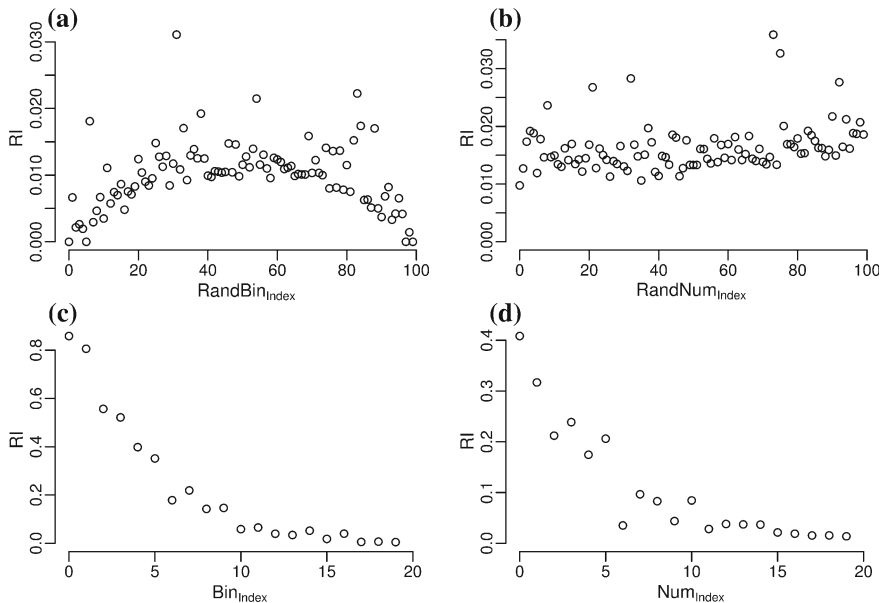


**Table 2** Results of MCFS on simulated data with 100 objects. Significant features and the three highest ranked non-significant features of each type are shown. The features are ranked according to their RI. Grayed lines denote non-significant features

Rank	Feature	RI	$\mu_{RIperm}$	$\sigma_{RIperm}$	$P_{z-test}$	$P_{perm-test}$
1	<i>Bin</i> <sub>0</sub>	0.859	0.0216	0.00458	<0.0001	0.0001
2	<i>Bin</i> <sub>1</sub>	0.806	0.0228	0.00600	<0.0001	0.0001
3	<i>Bin</i> <sub>2</sub>	0.557	0.0234	0.00548	<0.0001	0.0001
4	<i>Bin</i> <sub>3</sub>	0.521	0.0240	0.00519	<0.0001	0.0001
5	<i>Num</i> <sub>0</sub>	0.408	0.0344	0.00928	<0.0001	0.0001
6	<i>Bin</i> <sub>4</sub>	0.398	0.0248	0.01121	<0.0001	0.0001
7	<i>Bin</i> <sub>5</sub>	0.351	0.0239	0.00547	<0.0001	0.0001
8	<i>Num</i> <sub>1</sub>	0.317	0.0326	0.00694	<0.0001	0.0001
9	<i>Num</i> <sub>3</sub>	0.239	0.0334	0.00820	<0.0001	0.0001
10	<i>Bin</i> <sub>7</sub>	0.219	0.0251	0.00438	<0.0001	0.0001
11	<i>Num</i> <sub>2</sub>	0.212	0.0347	0.00941	<0.0001	0.0001
12	<i>Num</i> <sub>5</sub>	0.206	0.0344	0.01061	<0.0001	0.0001
13	<i>Bin</i> <sub>6</sub>	0.178	0.0249	0.00496	<0.0001	0.0001
14	<i>Num</i> <sub>4</sub>	0.174	0.0342	0.00715	<0.0001	0.0001
15	<i>Bin</i> <sub>9</sub>	0.146	0.0262	0.00462	<0.0001	0.0001
16	<i>Bin</i> <sub>8</sub>	0.142	0.0257	0.00660	<0.0001	0.0001
17	<i>Num</i> <sub>7</sub>	0.096	0.0343	0.00683	<0.0001	0.0019
18	<i>Num</i> <sub>10</sub>	0.084	0.0357	0.01153	<0.0001	0.0024
19	<i>Num</i> <sub>8</sub>	0.083	0.0353	0.01017	<0.0001	0.0032
20	<i>Bin</i> <sub>11</sub>	0.065	0.0271	0.00733	<0.0001	0.0020
21	<i>Bin</i> <sub>10</sub>	0.058	0.0277	0.00950	0.0006	0.0062
22	<i>Bin</i> <sub>14</sub>	0.052	0.0229	0.00486	<0.0001	0.0127
23	<i>Num</i> <sub>9</sub>	0.044	0.0361	0.00925	0.2058	0.1349
24	<i>Bin</i> <sub>16</sub>	0.040	0.0247	0.00546	0.0032	0.0418
25	<i>Bin</i> <sub>12</sub>	0.039	0.0274	0.00760	0.0572	0.0458
26	<i>Num</i> <sub>12</sub>	0.038	0.0343	0.00806	0.3310	0.2664
27	<i>Num</i> <sub>13</sub>	0.037	0.0336	0.00719	0.3127	0.2540
29	<i>RandNum</i> <sub>73</sub>	0.036	0.0326	0.00632	0.2990	0.1952
31	<i>Bin</i> <sub>13</sub>	0.034	0.0255	0.00784	0.1379	0.0829
32	<i>RandNum</i> <sub>75</sub>	0.033	0.0323	0.00645	0.4753	0.3798
33	<i>RandBin</i> <sub>31</sub>	0.031	0.0237	0.00647	0.1254	0.1010
34	<i>RandNum</i> <sub>32</sub>	0.028	0.0299	0.00605	0.6053	0.5193
39	<i>RandBin</i> <sub>83</sub>	0.022	0.0176	0.00522	0.1882	0.1670
41	<i>RandBin</i> <sub>54</sub>	0.021	0.0280	0.00857	0.7759	0.9115
64	<i>Bin</i> <sub>15</sub>	0.018	0.0244	0.01166	0.7145	0.9874

and permutation test. The threshold using the built-in test was 0.0352, which in this case identified the same features.

The relation between the RI scores and the indices of the features is shown in Fig. 3. There is a substantial decrease in the noise compared with using 100 objects.



**Fig. 2** Relation between attribute indices and RI using a dataset with 100 objects. Shown for **a, b** random and **c, d** informative features of both **a, c** binary and **b, d** numeric type. Note that the y-axis scale varies from panel to panel

### 3.2 Comparison of $p$ -Values

In order to determine how accurate the  $p$ -values obtained through the z-test were, we compared them with the permutation test  $p$ -values (Fig. 4a). Furthermore, we computed  $p$ -values based on the built-in method, and compared to the permutation test  $p$ -values (Fig. 4b).

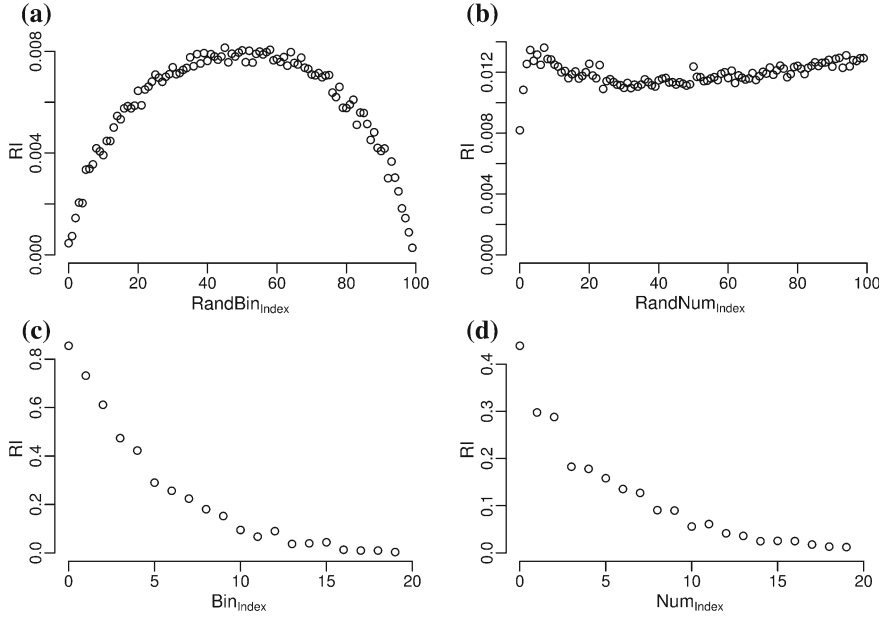
The  $p$ -values estimated using the z-test were closely following the ones obtained by permutation test, whereas the built-in method failed to efficiently model the empirical  $p$ -values, although the built-in method identified almost as many significant features as the z-test. Essentially, the  $p$ -values obtained by applying the built-in method were always equal to either 0 or 1. We speculate that the assumption of comparing two means results in a biased downward estimate of the variance of the data.

## 4 Discussion

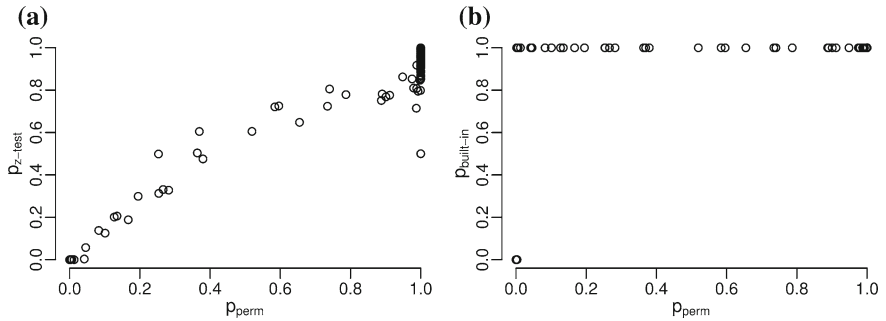
We have used simulated data to evaluate the application of a z-test to identifying features significant for classification using MCFS. The data was designed in such

**Table 3** Results of MCFS on simulated data with 1,000 objects. Significant features and the three highest ranked non-significant features of each type are shown. The features are ranked according to their RI. Grayed lines denote non-significant features

Rank	Feature	RI	$\mu_{RIperm}$	$\sigma_{RIperm}$	$P_{Z-test}$	$P_{perm-test}$
1	<i>Bin</i> <sub>0</sub>	0.855	0.0130	0.00126	<0.0001	0.0001
2	<i>Bin</i> <sub>1</sub>	0.732	0.0133	0.00134	<0.0001	0.0001
3	<i>Bin</i> <sub>2</sub>	0.612	0.0136	0.00139	<0.0001	0.0001
4	<i>Bin</i> <sub>3</sub>	0.474	0.0139	0.00149	<0.0001	0.0001
5	<i>Num</i> <sub>0</sub>	0.439	0.0316	0.00129	<0.0001	0.0001
6	<i>Bin</i> <sub>4</sub>	0.423	0.0139	0.00150	<0.0001	0.0001
7	<i>Num</i> <sub>1</sub>	0.298	0.0316	0.00122	<0.0001	0.0001
8	<i>Bin</i> <sub>5</sub>	0.290	0.0140	0.00162	<0.0001	0.0001
9	<i>Num</i> <sub>2</sub>	0.288	0.0316	0.00139	<0.0001	0.0001
10	<i>Bin</i> <sub>6</sub>	0.256	0.0147	0.00131	<0.0001	0.0001
11	<i>Bin</i> <sub>7</sub>	0.224	0.0147	0.00160	<0.0001	0.0001
12	<i>Num</i> <sub>3</sub>	0.183	0.0319	0.00149	<0.0001	0.0001
13	<i>Bin</i> <sub>8</sub>	0.180	0.0147	0.00155	<0.0001	0.0001
14	<i>Num</i> <sub>4</sub>	0.178	0.0320	0.00128	<0.0001	0.0001
15	<i>Num</i> <sub>5</sub>	0.158	0.0318	0.00108	<0.0001	0.0001
16	<i>Bin</i> <sub>9</sub>	0.152	0.0151	0.00141	<0.0001	0.0001
17	<i>Num</i> <sub>6</sub>	0.135	0.0317	0.00129	<0.0001	0.0001
18	<i>Num</i> <sub>7</sub>	0.127	0.0315	0.00134	<0.0001	0.0001
19	<i>Bin</i> <sub>10</sub>	0.095	0.0151	0.00165	<0.0001	0.0001
20	<i>Num</i> <sub>8</sub>	0.091	0.0316	0.00136	<0.0001	0.0001
21	<i>Num</i> <sub>9</sub>	0.090	0.0316	0.00147	<0.0001	0.0001
22	<i>Bin</i> <sub>12</sub>	0.090	0.0149	0.00145	<0.0001	0.0001
23	<i>Bin</i> <sub>11</sub>	0.067	0.0155	0.00141	<0.0001	0.0001
24	<i>Num</i> <sub>11</sub>	0.061	0.0312	0.00130	<0.0001	0.0001
25	<i>Num</i> <sub>10</sub>	0.056	0.0316	0.00153	<0.0001	0.0001
26	<i>Bin</i> <sub>15</sub>	0.044	0.0135	0.00134	<0.0001	0.0001
27	<i>Num</i> <sub>12</sub>	0.041	0.0314	0.00153	<0.0001	0.0001
28	<i>Bin</i> <sub>14</sub>	0.039	0.0145	0.00153	<0.0001	0.0001
29	<i>Bin</i> <sub>13</sub>	0.037	0.0153	0.00139	<0.0001	0.0001
30	<i>Num</i> <sub>13</sub>	0.036	0.0312	0.00149	0.0006	0.0046
31	<i>Num</i> <sub>15</sub>	0.025	0.0311	0.00131	1.0000	1.0000
32	<i>Num</i> <sub>16</sub>	0.025	0.0308	0.00115	1.0000	1.0000
33	<i>Num</i> <sub>14</sub>	0.025	0.0309	0.00118	1.0000	1.0000
35	<i>RandNum</i> <sub>7</sub>	0.014	0.0314	0.00134	1.0000	1.0000
36	<i>RandNum</i> <sub>3</sub>	0.013	0.0317	0.00077	1.0000	1.0000
38	<i>RandNum</i> <sub>5</sub>	0.013	0.0321	0.00103	1.0000	1.0000
40	<i>Bin</i> <sub>16</sub>	0.013	0.0125	0.00137	0.3742	0.4528
137	<i>Bin</i> <sub>18</sub>	0.010	0.0088	0.00089	0.1607	0.1239
138	<i>Bin</i> <sub>17</sub>	0.010	0.0106	0.00135	0.7737	0.8153
140	<i>RandBin</i> <sub>45</sub>	0.008	0.0156	0.00156	1.0000	1.0000
141	<i>RandBin</i> <sub>58</sub>	0.008	0.0150	0.00157	1.0000	1.0000
142	<i>RandBin</i> <sub>50</sub>	0.008	0.0157	0.00165	1.0000	1.0000



**Fig. 3** Relation between attribute indices and RI using a dataset with 1,000 objects. Shown for **a**, **b** random and **c**, **d** informative features of both **a**, **c** binary and **b**, **d** numeric type. Note that the y-axis scale varies from panel to panel



**Fig. 4** Agreement between the permutation test and **a**  $p$ -values obtained from  $z$ -test, or **b**  $p$ -values computed using the build-in strategy (showing upper-tail  $p$ -values). Calculated for the 100 objects dataset

a way that the influence of the distribution and domain of feature values could be evaluated. We have shown that the RI of a feature depends on its distribution of values across the objects. Features with more evenly distributed values tend to get higher RI scores. This is likely caused by the inclusion of the information gain in the calculation of the RI and may cause trouble if the RIs of all features are assumed to follow the same distribution.

The built-in test in the dmLab software assumes that the RI of all features derive from the same distribution, which may bias the estimate of the feature significances, essentially preventing some features from reaching significance if other—more favorably distributed—features are present in the data. In this study we suggest that each feature should be evaluated individually, using its own null model.

We have shown that a z-test efficiently estimates the correct  $p$ -value as validated by a permutation test, whereas applying the built-in strategy combining a t-test with a permutation test failed to detect some significant features and to estimate the “true”  $p$ -values obtained by the permutation test. The built-in-strategy treats the RI computed on the original data as a mean instead of a single observation, which may underestimate the sample variation.

It should be noted that since the true standard deviation and mean of the feature RIs on the permuted data is not known, at least 30 permutations have to be performed to convincingly estimate the distribution parameters from the observed data in order to apply a z-test. This puts a lower limit on the number of permutations that can be run to estimate the feature significances. The z-test requires the RIs measured for the permuted data to be approximately normally distributed. Almost all features in our study had a bell shaped distribution, but sometimes with an elongated tail in one direction. Such a tail may lead to an overestimation of the variance in the permuted data, underestimating the significance of a feature. However, we did not observe any such effect.

Since the features are scored according to how they participate in decision tree classifiers, non-informative features will generally not be selected when there are informative features in the same subset. Thus, the more informative features that are present in the data, the lower the non-informative features are scored. We do not expect this effect to significantly affect the estimated  $p$ -values of the informative features, but the, comparably, non-informative ones will get very poor  $p$ -values, which may explain why many features obtained  $p$ -values close to 1 using both the permutation test and the z-test.

Although this methodology is efficient at detecting informative features, the most significant features may not necessarily be the best features to use for classification. The effect size of a feature may be more important than its significance, and both the RI and the  $p$ -value should be considered when selecting features for classification.

## 5 Conclusions

MCFS is a reliable method for feature selection that is able to identify significant features, even with small effects. In this study we showed that features with more evenly distributed values tend to receive higher RIs than features with an uneven distribution. To avoid biasing the selection towards such features, each feature should be tested for significance separately. We have shown that a z-test is an efficient method to estimate the significance of a feature and that these  $p$ -values have a strong agreement with  $p$ -values obtained through a traditional permutation test.

**Acknowledgments** We wish to thank the reviewers for insightful comments that helped improve this paper. The authors were in part supported by an ESSENCE grant, by Uppsala University and by the Institute of Computer Science, Polish Academy of Sciences.

## References

1. Guyon I, Elisseeff A (2003) An introduction to variable and feature selection. *J. Mach. Learn. Res.* 3:1157–1182
2. Saey Y, Inza I, Larranaga P (2007) A review of feature selection techniques in bioinformatics. *Bioinformatics* 23:2507–2517
3. Draminski M, Rada-Iglesias A, Enroth S, Wadelius C, Koronacki J, Komorowski J (2008) Monte Carlo feature selection for supervised classification. *Bioinformatics* 24:110–117
4. Kierczak M, Ginalski K, Draminski M, Koronacki J, Rudnicki W, Komorowski J (2009) A rough set-based model of HIV-1 reverse transcriptase resistome. *Bioinform. Biol. Insights* 3:109–127
5. Draminski M, Kierczak M, Koronacki J, Komorowski J (2010) Monte Carlo feature selection and interdependency discovery in supervised classification. *Stud Comput Intell* 263:371–385
6. Enroth S, Bornelöv S, Wadelius C, Komorowski J (2012) Combinations of histone modifications mark exon inclusion levels. *PLoS ONE* 7:e29911
7. Bornelöv S, Sääf A, Melen E, Bergström A, Moghadam BT, Pulkkinen V, Acevedo N, Pietras CO, Ege M, Braun-Fahrlander C, Riedler J, Doekes G, Kabesch M, van Hage M, Kere J, Scheynius A, Söderhäll C, Pershagen G, Komorowski J (2013) Rule-based models of the interplay between genetic and environmental factors in Childhood Allergy. *PLoS ONE* 8(11):e80080
8. Kruczyk M, Zetterberg H, Hansson O, Rolstad S, Minthon L, Wallin A, Blennow K, Komorowski J, Andersson M (2012) Monte Carlo feature selection and rule-based models to predict Alzheimer's disease in mild cognitive impairment. *J Neural Transm* 119:821–831
9. <http://www.ipipan.eu/staff/m.draminski/files/dmLab185.zip>
10. Van AHT, Saey Y, Wehenkel L, Geurts P (2012) Statistical interpretation of machine learning-based feature importance scores for biomarker discovery. *Bioinformatics* 28:1766–1774
11. Damiński M, Kierczak M, Nowak-Brzezińska A, Koronacki J, Komorowski J (2011) The Monte Carlo feature selection and interdependency discovery is unbiased, vol 40, pp 199–211. Systems Research Institute, Polish Academy of Sciences

Challenges in Computational Statistics and Data Mining

Matwin, S.; Mielniczuk, J. (Eds.)

2016, X, 399 p. 73 illus., 3 illus. in color., Hardcover

ISBN: 978-3-319-18780-8