Manuel J. A. Eugster & Torsten Hothorn & Friedrich Leisch

Exploratory and Inferential Analysis of Benchmark Experiments

Technical Report Number 030, 2008
Department of Statistics
University of Munich

http://www.stat.uni-muenchen.de

Exploratory and Inferential Analysis of Benchmark Experiments

Manuel J. A. Eugster, Torsten Hothorn and Friedrich Leisch

Department of Statistics
Ludwig-Maximilians-Universität München
München, Germany
Firstname.Lastname@stat.uni-muenchen.de

Abstract: Benchmark experiments produce data in a very specific format. The observations are drawn from the performance distributions of the candidate algorithms on resampled data sets. In this paper we introduce a comprehensive toolbox of exploratory and inferential analysis methods for benchmark experiments based on one or more data sets. We present new visualization techniques, show how formal non-parametric and parametric test procedures can be used to evaluate the results, and, finally, how to sum up to a statistically correct overall order of the candidate algorithms.

Keywords: benchmark experiment, learning algorithm, visualisation, inference, ranking

1. Introduction

In statistical learning, benchmark experiments are empirical experiments with the aim of comparing and ranking algorithms with respect to a certain performance measure. New benchmark experiments are published on almost a daily basis; it is the primary method of choice to evaluate new learning algorithms in most research fields with applications related to learning algorithms. However, there are surprisingly few publications on *how* to evaluate benchmark experiments, some newer exceptions are Hothorn et al. (2005), Demsar (2006), Yildiz and Alpaydin (2006) and Hornik and Meyer (2007).

Hothorn et al. (2005) use the bootstrap as sampling scheme such that the resulting performance observations are independent and identically distributed (iid) and can be analyzed using standard statistical methods. However, their paper describes a general framework, not precise instructions for a concrete benchmark experiment. To use a metaphor, it describes how to cook in general, but contains no recipes for a nice dinner. Using the foundations layed out by the general framework, our goal is now to implement a toolbox of exploratory and inferential methods for the analysis of benchmark experiments.

This article is organized as follows: in Section 2 we review the design of a benchmark experiment following Hothorn et al. (2005). We also introduce an exemplar benchmark experiment which is used through out the article to illustrate our methods. In Section 3 we present the exploratory and inferential analysis of benchmark experiment based on one data set, and the establishment of an overall order based on different performance measures. Section 4 generalizes the benchmark experiment for the comparison of the candidate algorithms on more than one data set. As in the case of one data set, we present exploratory and inferential ways of analyses, and an overall order over all data sets and all performance measures, respectively. The article is concluded with a summary and an outlook for further developments in Section 5.

All computations are performed using R (R Development Core Team, 2008), the corresponding R functions are part of an R package for the analysis of benchmark experiments which is currently under development and will be released on CRAN in due course. Preliminary versions of the functions and all data used in this article are available from `http://www.statistik.lmu.de/~eugster/`.

2. Design of benchmark experiments

Following Hothorn et al. (2005), we set up a benchmark experiment according to their *real world situation*. Given is a data set $\mathfrak{L} = \{z_1, \dots, z_m\}$. We draw B learning samples using some resampling method, e.g. sampling with replacement (bootstrapping):

$$\begin{aligned}\mathfrak{L}^1 &= \{z_1^1, \dots, z_n^1\} \\ &\vdots \\ \mathfrak{L}^B &= \{z_1^B, \dots, z_n^B\}\end{aligned}$$

Another possibility is cross-validation. Furthermore we assume that there are $K > 1$ candidate algorithms a_k $(k = 1, \dots, K)$ available for the solution of the underlying problem. For each algorithm a_k the function $a_k(\cdot \mid \mathfrak{L}^b)$ is the fitted model based on the sample $\mathfrak{L}^b$. This function itself has a distribution $\mathcal{A}_k$ as it is a random variable depending on $\mathfrak{L}^b$:

$$a_k(\cdot \mid \mathfrak{L}^b) \sim \mathcal{A}_k(\mathfrak{L}),\ k = 1, \dots, K$$

The performance of the candidate algorithm a_k when provided with the training data $\mathcal{L}^b$ is measured by a scalar function p:

$$p_{kb} = p(a_k, \mathfrak{L}^b) \sim \mathcal{P}_k = \mathcal{P}_k(\mathfrak{L})$$

The p_{kb} are samples drawn from the distribution $\mathcal{P}_k(\mathfrak{L})$ of the performance measure of the algorithm k on the data set $\mathfrak{L}$.

In this paper, we illustrate the analysis of benchmark experiments by means of supervised learning problems. The observations z are of the form $z = (y, x)$ where y denotes the response variable and x describes a vector of input variables. The aim of this learning task is to construct a learner $\hat{y} = a_k(x \mid \mathfrak{L}^b)$ which, based on the input variables, provides us with information about the unknown response. The discrepancy between the true response y and the predicted response $\hat{y}$ for one observation z is measured by a scalar loss function $L(y, \hat{y})$. The above introduced performance measure p is in this case defined by some functional μ of the distribution of the loss function:

$$p_{kb} = p(a_k, \mathfrak{L}^b) = \mu(L(y, a_k(x \mid \mathfrak{L}^b))) \sim \mathcal{P}_k(\mathfrak{L})$$

For classification, common loss functions are the misclassification and the entropy. The missclassification error, for example, incur loss 1 if x is wrongly classified:

$$L(y, \hat{y}) = \begin{cases} 0 & y = \hat{y} \\ 1 & \text{otherwise} \end{cases}$$

For regression, the absolute error and the squared error are common loss functions. Both measure the amount by which the estimator differs from the quantity to be estimated. In case of the squared error, loss then incur quadtratic:

$$L(y, \hat{y}) = (y - \hat{y})^2$$

An adequate choice for the functional μ is the expectation E. The misclassification performance measure is then given by

$$p_{kb} = \mathsf{E}_{a_k} \mathsf{E}_{z=(y,x)} L(y, a_k(x \mid \mathfrak{L}^b))$$

with L the misclassification loss. Another conceivable choice of μ is the median, corresponding to the absolut loss. As, in most cases, we are not able to calculate μ analytically, we have to use the empirical analogue $\mu_{\mathfrak{T}}$ based on a test sample $\mathfrak{T}$:

$$\hat{p}_{kb} = \hat{p}(a_k, \mathfrak{L}^b) = \mu_{\mathfrak{T}}(L(y, a_k(x \mid \mathfrak{L}^b))) \sim \hat{\mathcal{P}}_k(\mathfrak{L})$$

For our analyses, independent observations of the performance measure are required. K-fold cross-validation results in dependencies between the different folds, therefore we use bootstrapping and define $\mathfrak{T}$ in terms of out-of-bootstrap observations: $\mathfrak{T} = \mathfrak{L} \setminus \mathfrak{L}^b$. Based on $\hat{p}_{kb}$, we estimate the empirical performance distribution $\hat{\mathcal{P}}_k(\mathfrak{L})$ for each candidate algorithm a_k and compare them with exploratory data analysis tools and formal inference procedures.

Example. To demonstrate our methods, we use an exemplar benchmark study. The primary goal of this example is to illustrate general interesting aspects of benchmark experiments; not necessarily using up-to-date classifiers. Because of the replaceability of the data set and the algorithms, we just introduce the setup and then encode them with letters and colors. The learning problem is the binary classification problem `monks3` (encoded with `I`) from the UCI Machine Learning repository (Asuncion and Newman, 2007). It consists of 6 nominal attributes and 554 observations. The candidate algorithms used are linear discriminant analysis (🟧, `orange`), naive bayes classifier (🟨, `yellow`), k-nearest neighbour classifier (🟪, `purple`), classification trees (🟥, `red`), support vector machines (🟦, `blue`), and neural networks (🟩, `green`) (all, e.g. Venables and Ripley, 2002; Hastie et al., 2001). Appendix B lists detailed information about the data set and the candidate algorithms. Misclassification error is used as performance measure, and the number of bootstrap samples B is 250.

The execution of this benchmark experiment results in 250 misclassification measures per candidate algorithm. These measures are the estimated empirical misclassification distributions of the candidate algorithms on data set `I` and are the base for the comparison of the algorithms right up to the arrangement of an order relation.

3. Analysis of a benchmark experiment

The first step is to analyse the benchmark experiment in an exploratory way. Based on findings in this step, the second step tests hypothesis of interest and yields a statistically correct order relation of the candidate algorithms.

3.1. Exploratory analysis

Common analyses of benchmark experiments consist of the comparison of the empirical performance measure distributions based on some summary statistics: algorithm a_u is better than algorithm a_v iff $\phi(\hat{\mathcal{P}}_u) < \phi(\hat{\mathcal{P}}_v)$. ϕ is a scalar functional and for example a measure of central tendency, statistical dispersion or shape. Additionally, confidence intervals can indicate the significance of differences.

$\phi =$	Mean	SD	Median	Max
`blue`	0.0110	0.0059	0.0100	0.0340
`red`	0.0116	0.0080	0.0100	0.0561
`green`	0.0293	0.0123	0.0273	0.0631
`yellow`	0.0344	0.0118	0.0340	0.0707
`purple`	0.0352	0.0094	0.0350	0.0561
`orange`	0.0353	0.0094	0.0350	0.0561

Table 1: Performance estimations based on common summary statistics ϕ: based on the B bootstrap samples, the mean, standard deviance (SD), median and maximum (Max) values of the empirical misclassification distributions are calculated.

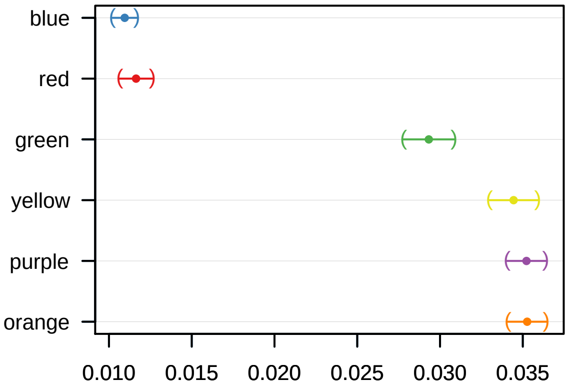


Figure 1: Visualisation of the 95% mean performance confidence intervals: the difference in the mean performances of two algorithm is not significant, if their intervals intersect.

Example (cont.). Table 1 shows the most established summary statistics for performance estimations. Based on the mean performance values (the other performance values are in an analogous manner), the order of the candidate algorithms is

$$\texttt{blue} < \texttt{red} < \texttt{green} < \texttt{yellow} < \texttt{purple} < \texttt{orange}.$$

The corresponding 95% confidence intervals are calculated by Mean $\pm\, 1.96 * \text{SD}/\sqrt{B}$ and shown in Figure 1. The confidence intervals of `blue`, `red` and `yellow`, `purple`, `orange` intersect, the differences between them are not significant.

Order relation. In cases of non-significant differences, we can not define a strict total order $<$ or a total order $\leq$ between the candidate algorithms. To model this circumstance, and to use further analyses methods in a consistent way, we define a reflexive and symmetric order relation $\approx$: two algorithms are $\approx$-related if their difference in a performance measure p is not significant.

Example (cont.). The candidates order based on the mean performance values then is

$$\texttt{blue} \approx \texttt{red} < \texttt{green} < \texttt{yellow} \approx \texttt{purple} \approx \texttt{orange}.$$

Worst-case scenario. A further analysis is based on the worst-case scenario and focus on the limitation of damage. Using the maximal performance values (Table 1, column Max), one can apply the *minimax rule* for minimizing the maximum possible loss. To make the minimax-order more robust against outliers, the m-worst performance values can be used; specify m, for example, according to the 95% quantiles of the performance distributions. To ignore very small differences in the values, one can define an ϵ-neighborhood in which all values are seen as equal. A more sophisticated approach is to additionally calculate the confidence intervals using bootstrapping.

Example (cont.). The order based on the worst performance values is

$$\texttt{blue} < \texttt{red} = \texttt{purple} = \texttt{orange} < \texttt{green} < \texttt{yellow}.$$

The m-worst values for $m = 12$ are

`blue`	`red`	`green`	`yellow`	`purple`	`orange`
0.0200	0.0202	0.0510	0.0545	0.0495	0.0495

The minimax-order with exact equality is:

$$\texttt{blue} < \texttt{red} < \texttt{purple} = \texttt{orange} < \texttt{green} < \texttt{yellow}$$

With the definition of a neighborhood where we ignore differences behind the third decimal place, i.e. $\epsilon = 0.001$, the difference between `blue` and `red` disappears:

$$\texttt{blue} = \texttt{red} < \texttt{purple} = \texttt{orange} < \texttt{green} < \texttt{yellow}$$

Basic plots. In many cases, these analyses based on the heavily compacted numbers of Table 1 are the only basis for a ranking of the algorithms. But in doing so, one loses a lot of interesting and primarily important information about the experiment. A first step towards a deeper analysis is the usage of common visualizations: *dot plots* (with the algorithms on the abscissa and their performances on the ordinate, represented with a dot for each benchmark run) allow the examination of the raw performances distributions; *box plots* summarize the performance distributions and allow the identification of outliers; *histograms* and *density estimates* as estimations of the unobservable underlying performance density function.

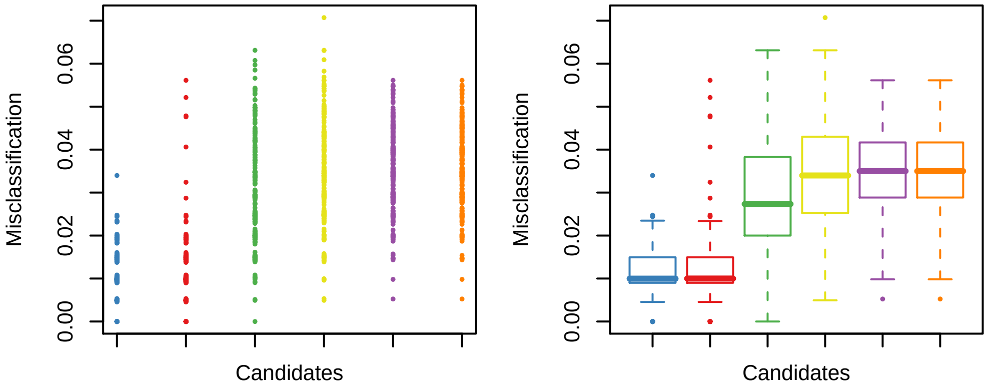


Figure 2: (a) Dot plot of the benchmark experiment result: the performance of each algorithm on each benchmark run is shown as a dot. (b) Box plot of the benchmark experiment result: the performance of each algorithm is aggregated by the five-number summary. Outliers are identified. In comparison to the dot plot, information about local minima is lost.

Example (cont.). Figure 2(a) and Figure 2(b) show the dot and box plot, respectively. Looking at the dot plot in, it can be seen that that the distributions for `blue` and `red` are not only skewed, but also multimodal. The algorithms often get stuck in local minima. The box plot supports this assumption, as we can see that the median is approximately equal to the first quantile. All other algorithms seems to be unimodal, and `green` slightly skewed.

The figures also allow an analysis of the overall order of the algorithms. `blue` and `red` have basically the same performance, the small differences in mean performance are caused mostly by a few outliers. Their performances somehow define the lower misclassification range from about 0 to 0.02. In this range, all algorithms find local minima with similar performance values, they present the same patterns. `green`, `yellow`, `purple` and `orange` mostly are in the upper misclassification range with some results in the lower one. `purple`, `orange` and `yellow` have the same performance, whereby the latter has some outliers including the worst performance at all. `green` has a huge variance. It has an outlier close to the best value of `blue` and `red`, and also results near to the worst performance.

One massive problem of the dot plot is the overdrawing of dots. For example, the impression of the order between `purple` and `yellow`; it seems that `purple` is better than `yellow`, but if we take a look at the summary statistics in Table 1, we see that the mean and median performances of `yellow` are slightly better. Additionally, the standard dot plot suggests the independence of the bootstrap samples. Indeed we know that, for example, `blue` and `red` perform similar over all benchmark runs, but we do not know their ranking per benchmark run, which algorithm is on which rank and how often. One possibility to avoid the first problem, is the usage of box plots. As we can see in Figure 2(b), the impression of the order is correct and it leads to similar conclusions, but we lose the information about the local minima. Another possibility is to jitter the dot plots, i.e., adding some random noise to the data. But both do not solve the second problem. The benchmark experiment plot was developed to overcome these limitations and to get a better understanding of benchmark experiments.

Benchmark experiment plot. Instead of random jittering, we use the ranks of the algorithms on each bootstrap sample to horizontally "stretch out" the dots. For each benchmark run, the algorithms are ordered according to their performance value: r_{ij} denotes the rank of p_{ij} in the joint ranking of $p_{i1}, \ldots, p_{iK}$, ties are broken at random. We draw separate dot plots for each rank. This can be seen as creating a "podium" with K places, and having separate dot plots for each podium place. The following pseudocode outlines the calculation of the benchmark experiment plot podium:

```
Input: $p_{ij}$ = matrix of performance values with $K$ columns and $B$ rows;
Output: $w_{ij}^k$ = list of $K$ podium places: each place is a matrix with $K$ columns and $B$ rows;
for $i = 1 \ldots B$ do
    for $j = 1 \ldots K$ do
        $r_{ij}$ = rank of $p_{ij}$ in the joint ranking of $p_{i1}, \ldots, p_{iK}$, ties are broken at random;
        $w_{i,j}^{r_{ij}} = p_{ij}$;
    end
end
```

Example (cont.). Figure 3 shows the benchmark experiment plot. It supports the assumption (given by Table 1 and Figure 2) that `blue` and `red` perform similar and significant better than all others. Both are almost equally often on place 1, but we see that `red` has some results on the

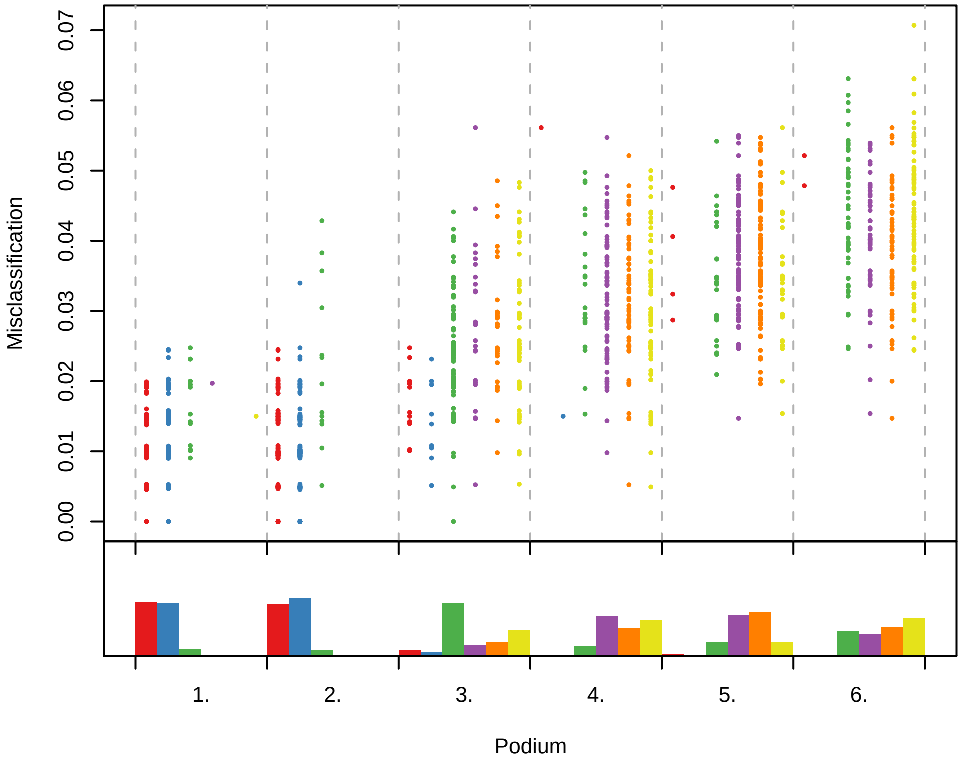


Figure 3: Benchmark experiment plot of the example: the abscissa is a podium with 6 places. For each benchmark run, the algorithms are sorted according to their performance values and a dot is drawn on the corresponding place. To visualise the count of an algorithm on a specific position, a bar plot is shown for each of podium places.

lower places. This is an aspect that is impossible to infer from the marginal distributions of the performance measures alone (regardless if displayed by dot plots, box plots, histograms, or density estimates). An example, where the impression of similar performances is not supported, is given in Eugster and Leisch (2008). According to the mean performance, `yellow` is on the fourth place, but we see that yellow has most last places. Similar things for `green`, even it is clearly the algorithm with most third places, it has its second-most results on the last one.

"Full" benchmark experiment plot. The "dots" in the displayed plots are not independent from each other, because all algorithms were evaluated on each bootstrap sample. This dependency can be displayed by connecting the dots corresponding to one bootstrap sample with a line, resulting in a modified version of a parallel coordinates plot.

In our implementation, the line segment between two podium places is drawn with the color of the algorithm in the lower position, to overcome the problem of overdrawing line we use transparency (alpha shading).

Example (cont.). In this "full benchmark experiment plot" one can also see correlations between algorithm performances (parallel vs. crossing lines). As an example, `red` and `blue` find the same local minima on almost all bootstrap samples. Another interesting fact between those two algo-

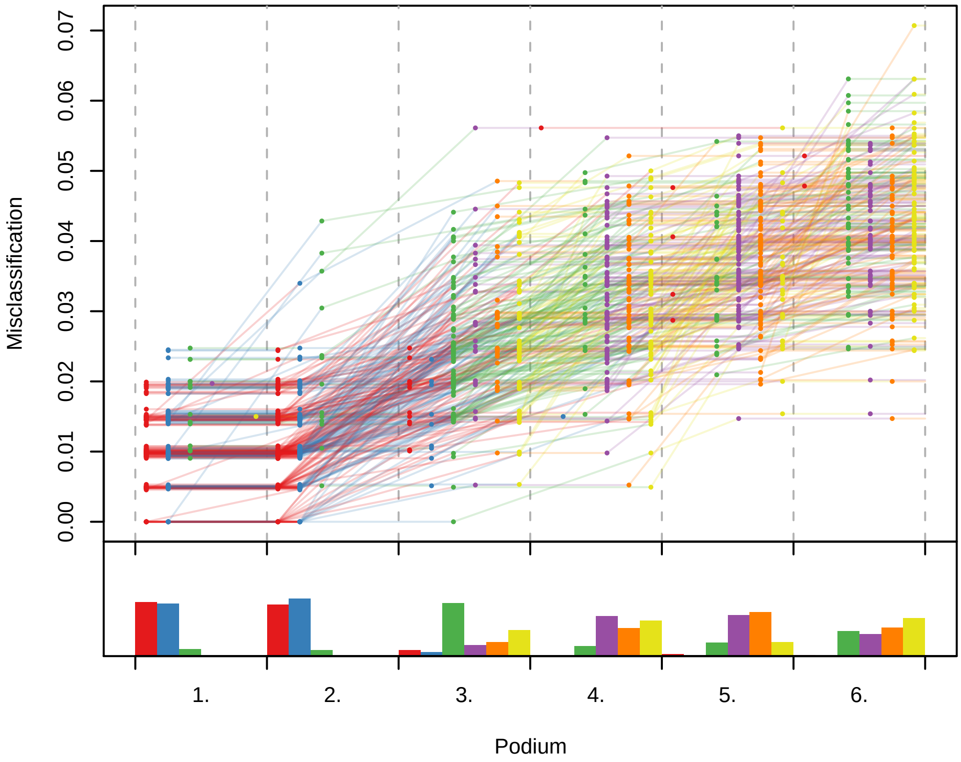


Figure 4: "Full" benchmark experiment plot: the benchmark experiment plot is extended to represent the dependency of the dots of one bootstrap sample with a line between them.

rithms is that whenever `red` performs well, `blue` performs well too: only horizontal lines going out of red dots in the first place. But this is not always true the other way round: there are non-horizontal lines going out of blue dots in the first place where red dots are even not on the second place. Further interesting analyses would explore these bootstrap samples and see if there are any noticeable characteristics which lead to this circumstance. The analysis of such coherences is simplified with an interactive version of the benchmark experiment plot with brushing techniques; our current work contains approaches in this direction.

The major goal of benchmark experiments is to derive an order of the candidate algorithms. Based on the two versions of the benchmark experiment plot we suggest:

$$\texttt{red} \approx \texttt{blue} < \texttt{green} < \texttt{yellow} \approx \texttt{orange} \approx \texttt{purple}.$$

3.2. Inference

To make a statistically correct order and ranking we need more formal tools; statistical inference and primarily the testing of hypothesis provides them. The design of a benchmark experiment is a random block design. This type of experiment has two classification factors: the experimental one, for which we want to determine systematic differences, and the blocking one, which represents a known source of variability. In terms of benchmark experiments, the experimental factor is the set of algorithms and the blocking factor is that all algorithms perform an the same bootstrap samples. The random block design is basically modelled by

$$p_{ij} = \kappa_0 + \kappa_j + b_i + \epsilon_{ij},$$
$$i = 1, \dots, B, \; j = 1, \dots (K-1),$$

with the set of candidate algorithms modelled as κ_j, and the sampling modelled as b_i. Now, different methods take different assumptions on κ_j, b_i and ϵ_{ij}. The null hypothesis is that of no algorithm differences,

$$H_0: \; \kappa_1 = \dots = \kappa_{K-1} = 0,$$
$$H_A: \; \exists j: \; \kappa_j \neq 0.$$

Common methods for testing the differences between the candidate algorithms are non-parametric (or distribution-free) tests.

Friedman test. A global test, whether there are any differences between the algorithms at all, can be performed using the Friedman test (e.g. Demsar, 2006; Hollander and Wolfe, 1999). This test takes the assumptions $\sum_{i=1}^{B} b_i = 0$, $\sum_{i=0}^{K-1} \kappa_j = 0$, ϵ_{ij} are mutually independent and each one comes from the same continuous population.

The Friedman procedure uses the ranks r_{ij} defined in the section about the benchmark experiment plot, but ties are averaged. Set $R_j = \sum_{i=1}^{B} r_{ij}$, $R_{.j} = \frac{R_j}{n}$, $R_{..} = \frac{K+1}{2}$ and compute the test statistic

$$S = \frac{12B}{K(K+1)} \sum_{j=1}^{K} (R_{.j} - R_{..})^2.$$

When H_0 is true, the statistic S has an asymptotic (B tending to infinity) χ^2 distribution based on $K-1$ degrees of freedom. We reject H_0, for a given significance level α, if $S \geq \chi^2_{K-1,\alpha}$. The distribution of the test statistic under the null hypothesis clearly depends on the (mostly) unknown distribution of the data and thus is (mostly) unknown as well. Hence we use permutation tests, where the unknown null distribution is replaced by the conditional null distribution, i.e., the distribution of the test statistic given the observed data (Hothorn et al., 2006).

Example (cont.). In case of our example benchmark experiment the exploratory analysis indicates that there are differences between the algorithms. Using this test, we can formally ensure that indications. The measures have been calculated as:

`blue`	`green`	`orange`	`purple`	`red`	`yellow`
R_1	R_2	R_3	R_4	R_5	R_6
393.5	955.5	1176.5	1175.5	413.5	1135.5

The test statistic S is 888.5657 and the p-value is $< 2.2 \times 10^{-16}$ (numerically zero). The test rejects the null hypothesis for all meaningful significance levels. We can proceed with a test with all pairwise comparisons and find the algorithms which actually differ.

Wilcoxon-Nemenyi-McDonald-Thompson test. This test is based on the Friedman's within-blocks ranks and is designed to make decisions about individual differences between pairs of treatments, i.e., algorithms (Hollander and Wolfe, 1999). The assumptions are the same as for the Friedman test.

Given the R_j, the procedure calculates the $K(K-1)/2$ differences $R_u - R_v$, $u = 1, \dots (K-1)$, $v = u, \dots, K$. At an experimentwise error rate α, the test then reaches its pairwise decisions, corresponding to each (κ_u, κ_v), by the criterion

$$\text{Decide } \kappa_u \neq \kappa_v \text{ if } |R_u - R_v| \geq r_\alpha\text{; otherwise decide } \kappa_u = \kappa_v,$$

where the constant r_α is chosen to make the experimentwise error rate equal to α. As before, the test is used in a permutation procedure.

Example (cont.). The measures $R_u - R_v$ haven been calculated as

		`yellow`	`red`	`purple`	`orange`	`green`
		R_6	R_5	R_4	R_3	R_2
`blue`	R_1	742	20	782	783	562
`green`	R_2	180	−542	220	221	
`orange`	R_3	−41	−763	−1		
`purple`	R_4	−40	−762			
`red`	R_5	722				

and the corresponding p-values

	`yellow`	`red`	`purple`	`orange`	`green`
`blue`	0.000	0.996	0.000	0.000	0.000
`green`	0.000	0.000	0.000	0.000	
`orange`	0.907	0.000	1.000		
`purple`	0.916	0.000			
`red`	0.000				

The null hypothesis of no difference is rejected for all pairs of candidate algorithms, except (`blue`, `red`), (`orange`, `purple`), (`orange`, `yellow`) and (`purple`, `yellow`).

Topological sort. Between pairs with significant differences we can establish a strict total order $<$, pairs with non-significant differences are $\approx$-related. An overall order of the algorithms is then defined using a topological sort. In general, a topological sort describes the sequence of elements with predefined dependencies accomplished (e.g., Knuth, 1997). In our case, the dependencies are the strict total orders between pairs of algorithms defined through the pairwise tests.

Example (cont.). The topological sort of the pairwise test results of the Wilcoxon-Nemenyi-McDonald-Thompson procedure is:

$$\texttt{blue} \approx \texttt{red} < \texttt{green} < \texttt{orange} \approx \texttt{purple} \approx \texttt{yellow}.$$

Nemenyi-Wilcoxon-Wilcox-Miller test. The above test compares all algorithms against each other, i.e., $\binom{K}{2}$ comparisons. Another scenario is the existence of a reference algorithm where we compare some algorithms with, i.e., $(K-1)$ comparisons. An example for this situation is a present reference implementation of method and some new implementations in different programming languages. The non-parametric procedure of choice is the Nemenyi-Wilcoxon-Wilcox-Miller test, we refer to Hollander and Wolfe (1999) for the concrete definition.

Mixed effects model. Using these non-parametric tests we indeed get answers for our hypothesis of interest, i.e., the p-values, but we do not really model the random block design and thus get no estimates for the parameters κ_j, b_i and ϵ_{ij}. We are not aware of any non-parametric method applicable in this situation. However, the parametric mixed effects models (e.g. Pinheiro and Bates, 2000) do, and since we are able to draw as many random samples B from the performance distributions as required, we can rely on asymptotic normal theory. κ_j is now a fixed, b_i a random effect. The assumptions are $b_i \sim N(0, \sigma_b^2)$ and $\epsilon_{ij} \sim N(0, \sigma^2)$. Hence, we estimate only one parameter σ_b^2 for the effect of the data set. A modelling, by contrast, with the effect of the data set as main effect, would have lead to B parameters.

The most common method to fit the linear mixed effects model is to estimate the "variance components" by the optimization of the restricted maximum likelihood (REML) through EM iterations or through Newton-Raphson iterations (see e.g. Pinheiro and Bates, 2000).

Example (cont.). The estimates for the parameters have been calculated as

$$\hat{\sigma}_b = 0.0052, \hat{\sigma} = 0.0082$$

and

Intercept (`blue`)	Δ`green`	Δ`orange`	Δ`purple`	Δ`red`	Δ`yellow`
$\hat{\beta}_0$	$\hat{\beta}_1$	$\hat{\beta}_2$	$\hat{\beta}_3$	$\hat{\beta}_4$	$\hat{\beta}_5$
0.0110	0.0184	0.0243	0.0243	0.0007	0.0235

where Δ denotes the difference between the Intercept and the corresponding algorithm. The global test, whether there are any differences between the algorithms which do not come from the sampling, can be performed with ANOVA and the F-test. For our model this test rejects the null hypothesis that all algorithms have the same performance with a p-value $< 2.2 \times 10^{-16}$. This is the equivalent to the non-parametric test procedure with a reference algorithm.

To test for pairwise differences, we use *Tukey contrasts.* As a major advantage compared to the non-parametric methods, can calculate simultaneous confidence intervals (Hothorn et al., 2008). Figure 5 shows the corresponding 95% family-wise confidence intervals. The differences between (`red`, `blue`), (`purple`,`orange`), (`yellow`,`orange`) and (`yellow`,`purple`) are not significant, the corresponding confidence intervals intersect zero and overlap each other. In analogous manner as described in the non-parametric case, we establish an algorithm order:

$$\texttt{blue} \approx \texttt{red} < \texttt{green} < \texttt{orange} \approx \texttt{purple} \approx \texttt{yellow}.$$

The order is equivalent to the non-parametric Wilcoxon-Nemenyi-McDonald-Thompson test, indicating the validity of the assumptions made when applying the linear mixed effects model.

3.3. Overall analysis

The exploratory and inferential analyses lead to different orders based on different standards of knowledge and questions of interests. We now use ranking and order mechanisms to sum up and gain an overall analysis.

Example (cont.). Amongst others, the last two sections led to the following most interesting orders: R_w, based on the minimax rule with $m = 12$,

$$\texttt{blue} < \texttt{red} < \texttt{purple} = \texttt{orange} < \texttt{green} < \texttt{yellow},$$

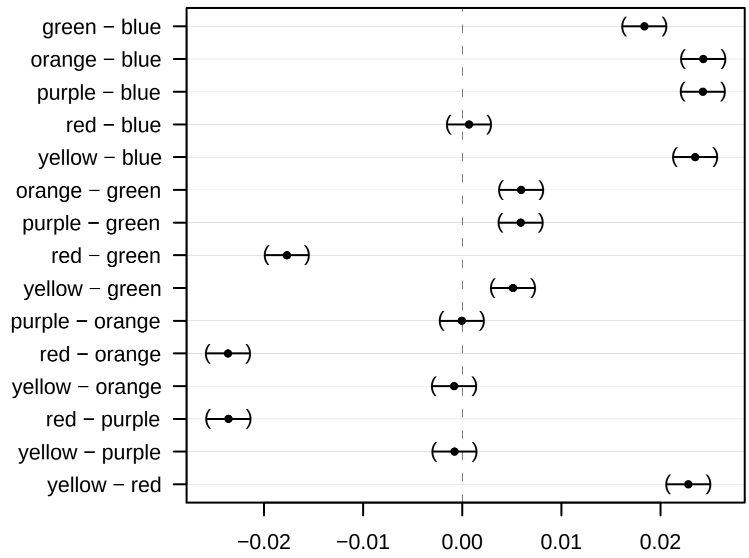


Figure 5: Simultaneous 95% confidence intervals for multiple comparisons of means using Tukey constrast based on the mixed effects model of the example experiment.

and R_m, based on the mixed effects model,

$$\texttt{blue} \approx \texttt{red} < \texttt{green} < \texttt{orange} \approx \texttt{purple} \approx \texttt{yellow}.$$

The minimax order as estimation of the worst performance and the mixed effects model order as estimation of the mean performance. Additionally we consider the computation time. Figure 6 shows the mean computation times of the candidate algorithms, which leads to the following order R_c:

$$\texttt{red} < \texttt{purple} < \texttt{orange} < \texttt{yellow} < \texttt{green} < \texttt{blue}$$

It is possible indeed to consider others like memory requirements, personal preferences or any other performance measure p. A lot of possibilities exist to create an overall analysis, we introduce two methods, the hierarchical order and the consensus ranking.

Hierarchical order. The hierarchical order h is a simple mechanism: starting from a base order, ties are broken by further ones. So, the sequence of the considered orders matters.

Example (cont.). Suppose, that we want an overall ranking where the mean performance is most important, then comes the worst performance and least important is the computation time, i.e., $h(R_m, R_w, R_c)$. Starting from R_m, the first step uses R_w to break the ties `blue` $\approx$ `red` and `yellow` $\approx$ (`orange` $\approx$ `purple`). The second step uses R_c to break the last tie `orange` $\approx$ `purple`. The resulted hierarchical order is:

$$\texttt{blue} < \texttt{red} < \texttt{green} < \texttt{purple} < \texttt{orange} < \texttt{yellow}$$

Consensus. A method where all orders are equally relevant is the consensus. In general the consensus is the aggregation of preferences of voters, Hornik and Meyer (2007) adapt this for

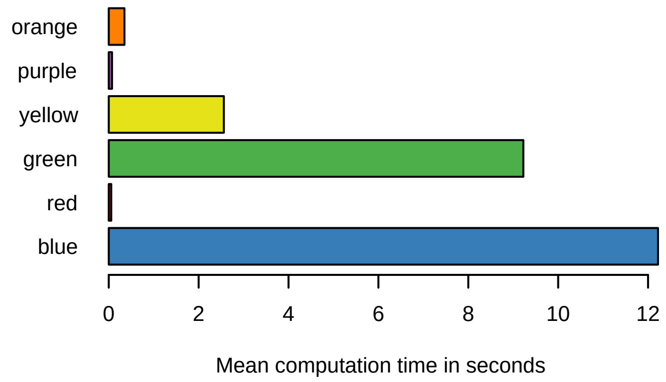


Figure 6: Mean computation time of the candidate algorithms in seconds. The benchmark experiment run on a workstation with a AMD Sempron 3400+ (2.00 gigahertz) processor and 1 gigabyte main memory.

application on benchmark experiments featuring more than one data set. We take up the idea, but instead of different data sets we apply on different performance measures. The consensus ranking R is a suitable aggregation of an ensemble of relations $\{R_1, \dots, R_r\}$. Hornik and Meyer (2007) present different methods, so-called constructive, axiomatic and optimization approaches. As the optimization approach formalizes the natural idea of describing consensus relations, we apply this one. This approach minimizes a criterion function

$$L(R) = \sum_{i=1}^{r} w_i d(R, R_i),$$

where R is element of a class C of admissible consensus relations. This class C describes the requirements on the ranking, e.g. a linear order (complete, antisymmetric, and transitive) or a complete preorder (complete and transitive). d is the symmetric difference distance between two relations, i.e. the cardinality of the symmetric difference of the relations, or equivalently, the number of pairs of algorithms being in exactly one of the two relations. w_i is the case weight given to R_i. For the concrete solution, this problem is reformulated as an integer program, we refer to the original publication for details.

Example (cont.). We have the relations $\{R_m, R_w, R_c\}$, as class C we define the family of linear orders. The ranking may not be unique, for example the consensus based on R_m and R_w results in 6 relations with minimal criterion $L(R)$:

`blue` < `red` < `green` < `orange` < `purple` < `yellow`
`blue` < `red` < `green` < `purple` < `orange` < `yellow`
`blue` < `red` < `orange` < `green` < `purple` < `yellow`
`blue` < `red` < `orange` < `purple` < `green` < `yellow`
`blue` < `red` < `purple` < `green` < `orange` < `yellow`
`blue` < `red` < `purple` < `orange` < `green` < `yellow`

If we take a look at R_m and R_w we can explain the result. The ties `blue` ≈ `red` and `yellow` ≈ (`orange` ≈ `purple`) in R_m are broken by their fix order in R_w. This results in the fix positions of these three algorithms. The different combinations of the remaining three algorithms yield through the fact that no order breaks the tie between `orange` and `purple`, and `green` is in one order in front and in the other order behind them. As next step we take the computation time into account, the consensus ranking reduces to two relations:

$$\texttt{red} < \texttt{blue} < \texttt{purple} < \texttt{orange} < \texttt{green} < \texttt{yellow}$$
$$\texttt{blue} < \texttt{red} < \texttt{purple} < \texttt{orange} < \texttt{green} < \texttt{yellow}$$

The different combinations from the previous consensus are now clarified, but with the computation time a new order in respect to `blue` and `red` is introduced which leads to the two relations. To gain a unique solution we weight the relations, e.g. we decide that worst case performance is more important than computation time and mean performance ($w_m = 1$, $w_w = 1.2$ and $w_c = 1$); the overall analysis according to our requirements then provides the ranking:

$$\texttt{blue} < \texttt{red} < \texttt{purple} < \texttt{orange} < \texttt{green} < \texttt{yellow}$$

4. Analysis of a benchmark experiment with more than one data set

Besides the analysis of a set of candidate algorithms on one data set, the extension to a set of data sets is obvious. The benchmark experiment then features M data sets $\mathcal{D} = \{\mathfrak{L}_1, \dots, \mathfrak{L}_M\}$. Clearly, the analysis depends on the specific collection of data sets and is primarily conditional on $\mathcal{D}$. But even with no algorithm being able to uniformly outperform all others for all possible data sets, this analysis still is interesting within a small problem domain. For the sake of clearness, we declare a *benchmark survey* as a benchmark experiment with more than one data set.

Example (cont.). For illustration purposes we extend the exemplar benchmark experiment introduced in the previous section to a benchmark survey. $\mathcal{D}$ is made up of 21 binary classification problems originated from the UCI Machine Learning repository (Asuncion and Newman, 2007). See Appendix B.2 for the concrete data sets, their descriptions and encodings. Assume that for each of the M data sets a benchmark experiment and an analysis as shown in the last section is made. We rely the following analyses on mixed effects models, the usage of non-parametric methods instead is analogue.

4.1. Exploratory analysis

Trellis displays (Becker et al., 1996) can give a rough overview of such a benchmark surveys. They contain one or more panels, arranged in a regular grid-like structure. For each data set one panel is used and a certain type of graph is displayed. As example, Figure 7 shows a trellis plot with box plots. But, actually, we have more information then the raw performance measures, we have an analysis of each single benchmark experiment corresponding to the last section. We can use these information to display a more specific picture of the benchmark survey.

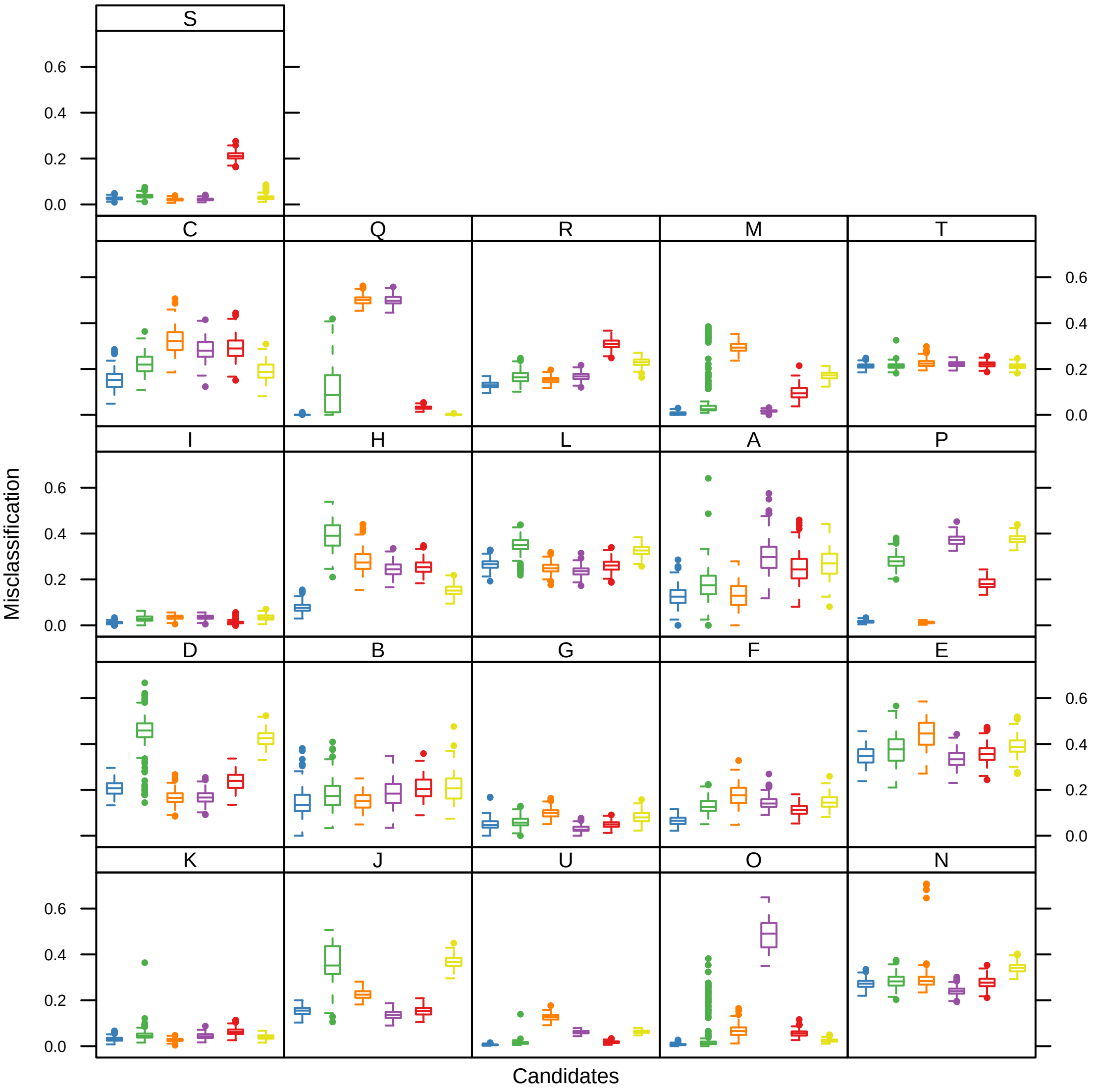


Figure 7: Grouped box plots (Trellis display): the raw performance measures of the candidate algorithms are grouped by the data sets and aggregated by five-number summaries. Each five-number summary is displayed as a box plot in a panel.

Benchmark survey plot. Let $\{R_1 \ldots, R_{\mathtt{M}}\}$ be the set of candidate algorithm orders based on linear mixed effects models. The benchmark survey plot visualizes the orders and an additional summary statistic of the benchmark experiments in a regular grid-like structure. The x-axis represents the data sets, the y-axis the podium. For each data set $\mathfrak{L}_i$, the podium places are painted according to the corresponding order R_i with “light” colors. Additionally, performance estimates are shown as percental “dark” bars and the significant difference between two algorithms is shown with a black border. In case of equivalent algorithms, the ties are broken using the summary statistic,

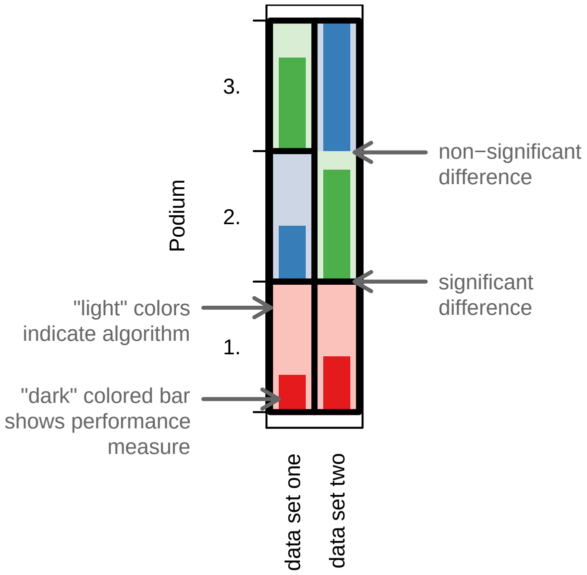

Podium
3.
2.
1.
non−significant difference
significant difference
"light" colors indicate algorithm
"dark" colored bar shows performance measure
data set one
data set two

Figure 8: Explanation of the benchmark survey plot idea: the abscissa represents the data sets, the ordinate the podium. Podium places are colored according to the corresponding algorithms, the bars show performance estimates. A significant difference between two algorithms is shown with a black border, which is missing in case of non-significance.

but their non-significant difference is shown with the missing of the black border between them. See Figure 8 for a visual explanation of the benchmark survey plot idea.

For the x-axis we want “similar” data sets close together. We determine the order of the data sets by the similarity of the corresponding orders of the candidate algorithms. Concerted to the similarity measure d used for the consensus ranking, we define similarity in terms of the symmetric difference distance, too. The similarity of the order relations is then determined by the hierarchical clustering of the distance matrix, and the data set order on the x-axis of the benchmark survey plot corresponds to the leafs of the resulting dendrogram.

Example (cont.). Figure 9 shows the benchmark survey plot in case of our example survey with the mean performance as dark bars. Table 2 shows the distance matrix, and Figure 10 the appropriate dendrogram computed by complete-linkage clustering.

The benchmark survey plot shows that `blue` has most first places (13 times) and is never worse than a third place. `purple` has second most first places (8 times), some last places, and it is the algorithm with the highest performance value (on data set `0`). `orange` has 6 first places, and is the algorithm with most last places (9 times). `yellow`, `red` and `green` have one or two first places. Besides that, they dominate the middle places, whereas `red` seems to be better than `green`, and `green` seems to be better than `yellow`.

Benchmark survey graph. Another representation method of the benchmark survey based on the distance matrix is a graph. The graph is a complete one, i.e., every pair of distinct vertices is connected by an edge. The vertices represent the data sets, the edges the distance between the data sets. Both graph elements can be used to show additional information, e.g., fill vertices with the colors of the winner algorithms.

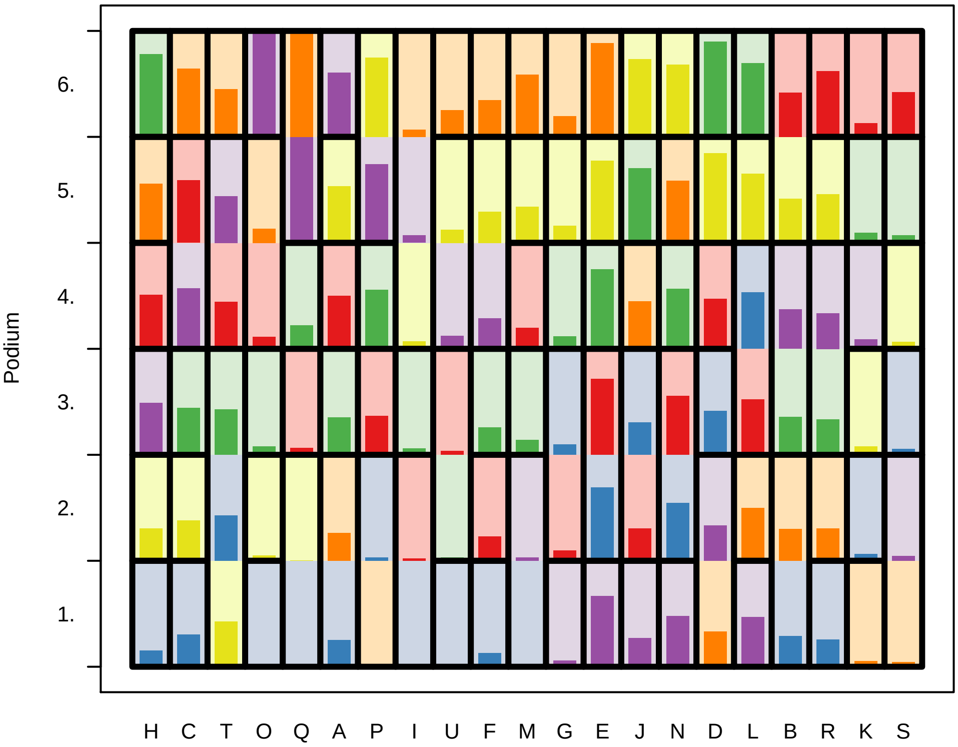


Figure 9: Benchmark survey plot: this plot visualizes the order relations of the candidate algorithms (the podium on the ordinate) on all data sets (the abscissa). The podium places are painted in the corresponding colors of the algorithms, significance is shown with black borders. An additional performance measure (e.g., the mean misclassification) is displayed with percental bars.

Example (cont.). Figure 11 shows the graph in case of the exemplar benchmark survey. The layout of the graph is calculated with the Kamada-Kawai algorithm implemented in the Graphviz software (Gansner and North, 2000). To avoid a too complex and unclear graph, we only show the first 15 of 24 distance levels with different colors and line widths (from broad dark green to strait light grey). Vertices whose data sets have a unique winner are filled with the color of the algorithm. If there are more then one first-placed algorithms, vertices are not colored.

The graph shows kind of two territories: right from the `K` node, there is the `blue` territory and left there is the `purple` territory. In combination with the benchmark survey plot in Figure 9, we see that in their territory the corresponding algorithm performs best on each data set. This information can be used, for example, in a problem domain to chose “the best” algorithm for a subset of problems or for some meta analyses.

4.2. Consensus

Hornik and Meyer (2007) determine an overall order by the derivation of a consensus ranking based on the individual benchmark data sets. We used the consensus method in section 3.3 to aggregate different performance measures, now we use it in the original sense and derive a ranking over the M data sets and based on a specific performance measure.

	K	J	U	O	N	D	B	G	F	E	I	H	L	A	P	C	Q	R	M	T
S	5	12	21	16	13	4	8	16	22	16	21	13	6	14	15	16	19	8	14	19
T	16	19	8	7	16	23	17	15	9	15	12	10	23	15	18	3	6	15	11	
M	17	8	7	14	5	14	12	4	8	4	11	11	14	16	17	10	15	12		
R	7	12	15	12	13	12	2	16	16	16	15	13	14	8	9	14	17			
Q	16	19	8	5	18	19	19	15	7	15	8	8	21	15	14	5				
C	13	18	7	4	15	20	16	14	8	14	11	7	20	14	17					
P	12	13	10	13	16	11	7	17	9	17	8	16	13	3						
A	9	16	11	10	17	14	6	20	12	20	11	17	16							
L	11	6	19	22	9	2	12	10	18	10	17	13								
H	10	11	12	11	14	13	15	11	11	11	14									
I	20	11	4	11	10	17	15	9	3	9										
E	21	4	9	18	3	12	16	0	8											
F	21	12	1	10	11	18	16	8												
G	21	4	9	18	3	12	16													
B	7	12	15	14	13	10														
D	9	8	19	20	11															
N	18	3	10	19																
O	11	22	9																	
U	20	13																		
J	17																			

Table 2: Pairwise distances between the data sets: the distance between two data sets is the symmetric difference distance between the corresponding order relations of the candidate algorithms.

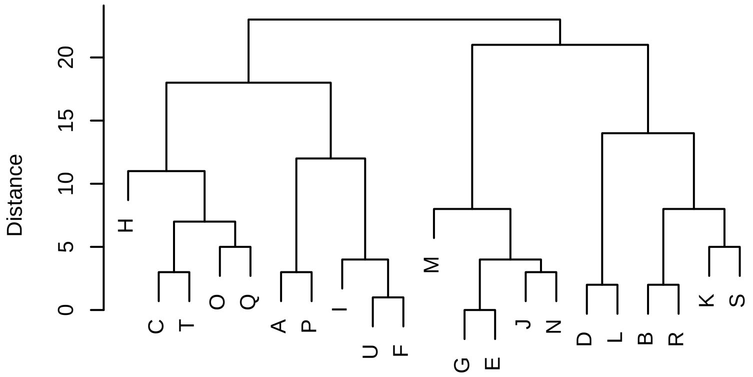


Figure 10: The dendrogram displays the similarity of the data sets based on the distance matrix in Table 2. It is estimated by hierarchical cluster analysis using the complete-linkage method.

Example (cont.). Let, as before, $\{R_{\mathtt{A}}, \ldots, R_{\mathtt{U}}\}$ be the set of candidate algorithm orders based on linear mixed effects models. The consensus (with C the family of linear orders) results in two relations with minimal criterion $L(R)$:

$$\texttt{blue} < \texttt{purple} < \texttt{red} < \texttt{green} < \texttt{orange} < \texttt{yellow}$$
$$\texttt{blue} < \texttt{purple} < \texttt{red} < \texttt{green} < \texttt{yellow} < \texttt{orange}$$

The two relations indicate that `orange` and `yellow` perform equal and the overall ranking based on the consens is

$$\texttt{blue} < \texttt{purple} < \texttt{red} < \texttt{green} < \texttt{yellow} \approx \texttt{orange}.$$

By means of the benchmark survey plot in Figure 9, the consensus is a sum up along the x-axis, whereat only the background colors (i.e., the algorithms) count and the bars (i.e., the mean performances) are not heeded.

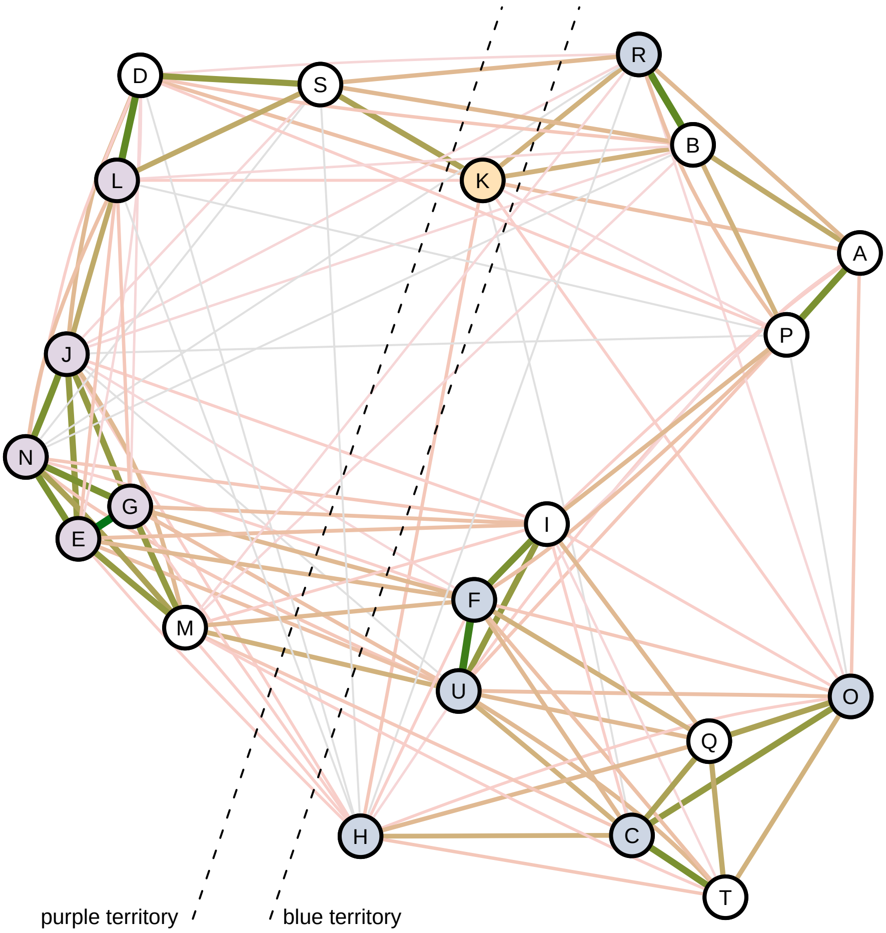


Figure 11: Benchmark survey graph: another representation of the distance matrix. The first 15 distance levels are shown, the color and the with of the edges represent them. Vertices are filled according to the winner algorithm if there is a unique one.

4.3. Inference

The design of a benchmark survey is a design with two experimental factors and their interactions, and blocking factors at two levels (e.g. Pinheiro and Bates, 2000). It is modelled by

$$p_{ijk} = \kappa_0 + \kappa_j + \gamma_k + \delta_{jk} + b_k + b_{ki} + \epsilon_{ijk}$$
$$i = 1, \ldots, B,\ j = 1, \ldots, (K-1),\ k = 1, \ldots, (M-1),$$

with the set of algorithms modelled as κ_j, the set of data sets as γ_k, and the interactions between algorithms and data sets as δ_{jk}. The random effects of the data sets are modelled as b_k, the sampling within the data sets as b_{ki} and the systematic error as ϵ_{ijk}. Similar to the case of on data set, the general hypothesis is that of no algorithm differences,

$$H_0: \ \kappa_1 = \cdots = \kappa_{K-1} = 0,$$
$$H_A: \ \exists j: \ \kappa_j \neq 0.$$

Mixed effects model. With linear mixed effects models, the assumptions on the random effects are $b_i \sim N(0, \sigma_1^2)$, $b_{ik} \sim N(0, \sigma_2^2)$ and $\epsilon_{ijk} \sim N(0, \sigma^2)$. Therefore, we estimate two parameters σ_1^2 and σ_2^2 for the effect of the data sets and the sampling within the data sets, respectively. Additionally, $1 + (K - 1) + (M - 1) + (K - 1)(M - 1)$ fixed effects parameters are estimated.

Example (cont.). In case of our exemplar benchmark survey these are $3 + 126$ parameters, see Appendix A.2 for the parameters and a model summary. The global test with ANOVA and the F-test rejects the null hypothesis that all algorithms have the same performance on all data sets. Using Tukey contrasts we test pairwise differences and calculate simultaneous confidence intervals. Figure 12 shows the 95% family-wise confidence intervals. The only Non-significant difference is between `orange` and `green`). An interesting aspect appears, `blue` is a lot better than all other algorithms. We establish the algorithm order

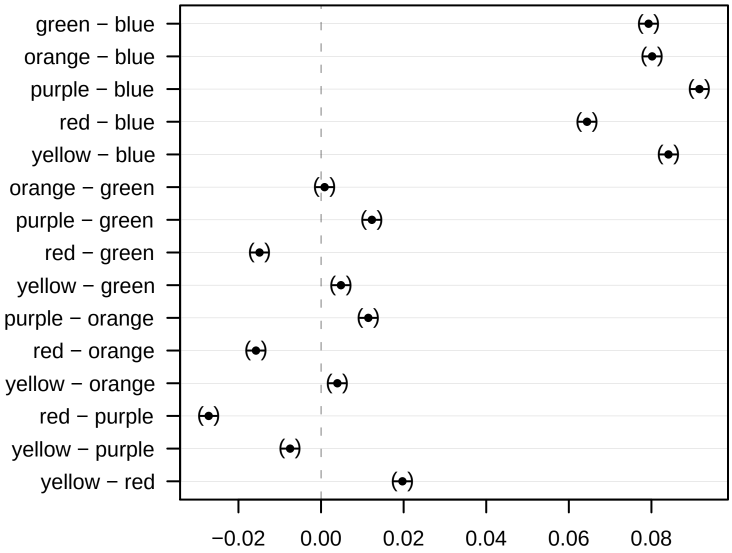


Figure 12: Simultaneous 95% confidence intervals for multiple comparisons of means using Tukey constrast based on the mixed effects model of the example experiment.

$$\texttt{blue} < \texttt{red} \approx \texttt{orange} \approx \texttt{green} < \texttt{yellow} < \texttt{purple}.$$

By means of the benchmark survey plot in Figure 9, this order mechanism heeds the background colors (i.e., the algorithms) and the bars (i.e., the mean performances). This explains the differences between this and the consensus order. As example, `purple` is second-ranked by the consensus ranking and last-ranked by this ranking. The reason is that `purple` performs well only if all other algorithms perform well too. In case of data set `E` wins `purple` against `green`, but only with a small mean performance difference; in case of data set `Q` is `green` the winner against `purple` and the mean performance difference is huge. This pattern is visible through all comparisons, and with the consensus ranking it is not possible to model this circumstance.

4.4. Overall analysis

In a similar manner to the overall analysis of a benchmark experiment, an overall analysis of a benchmark study is possible too. Again, one can sum up orders based on different performance measures with the mechanisms introduced in section 3.3.

5. Summary and future work

In this paper we present exploratory and inferential analyses of benchmark experiments based on one or more data sets. We introduce several new visualisation methods, the benchmark experiment plot, the benchmark survey plot and the benchmark survey graph. Parametric and non-parametric procedures are introduced to formally analyze benchmark experiments and surveys. In case of the benchmark experiment the design is a random block design and we show non-parametric procedures based on the Friedman test, and model the design using mixed effects, a parametric approach. In case of the benchmark survey the design has two experimental factors, their interactions and blocking factors at two levels; we model it using mixed effects. The formal procedures allow to test various hypothesis of interest, amongst others, the pairwise differences. We introduce an order relation for algorithms with non-significant differences, and infer a statistically correct order of the candidate algorithms, even in cases with more than one performance measure.

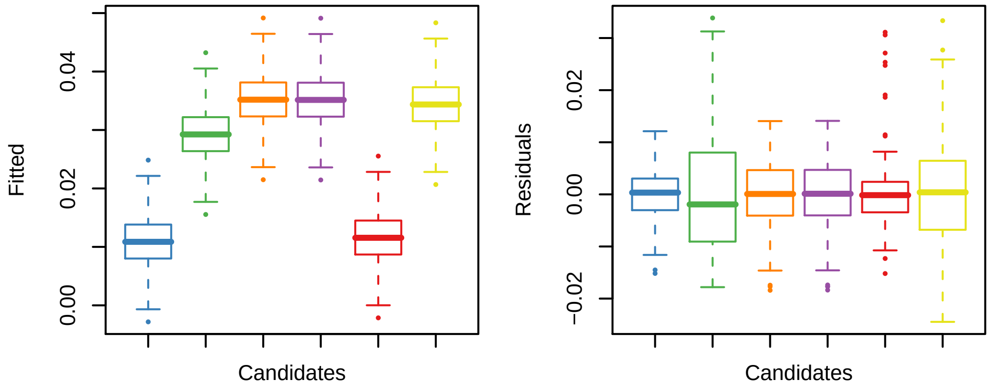
Future work contains work in all three (abstract) levels of the benchmarking process: Setup, Execution and Analysis. We want to use sequential testing to reduce computation time and study other modelling mechanism like mixture and hierarchical models. Furthermore, we want to examine the methodology of benchmarking and see if there are even any differences if one replaces a element of the benchmarking process by another one, e.g., uses the parametric test procedures instead of the non-parametric ones.

Acknowledgement

The authors want to thank Sebastian Kaiser and Fabian Scheipl for discussions and ideas.

A. Summary of the fitted mixed effects models

A.1. Benchmark experiment with data set I



(a)

(b)

Figure 13: Mixed effects model of the exemplar benchmark experiment: (a) box plot of the fitted values and (b) box plot of the residuals.

In section 3.2 we fit a linear mixed effects model for the inferential analysis of the exemplar benchmark experiment. Figure 13(a) shows a box plot of the fitted values grouped by the candidate algorithms. In comparison with Figure 2(b), the box plot of the raw performance values, no serious differences are visible. Figure 13(b) shows a box plot of the residuals. All mean values are near zero and except algorithm `green` all deviations are symmetric.

A.2. Benchmark survey with data sets A, ..., U

In section 4.3 we fit a linear mixed effects model for the inferential analysis of the exemplar benchmark experiment with more than one data set. The estimates for the model parameters have been calculated as

$$\hat{\sigma}_1 = 0.0015, \hat{\sigma}_2 = 0.0130, \hat{\sigma} = 0.0349$$

and

Intercept	Δgreen	Δorange	Δpurple	Δred	Δyellow	ΔJ	ΔU
β_0	β_1	β_2	β_3	β_4	β_5	γ_1	γ_2
0.0310	0.0173	−0.0040	0.0134	0.0330	0.0090	0.1223	0.0248
ΔO	ΔN	ΔD	ΔB	ΔG	ΔF	ΔE	ΔI
γ_3	γ_4	γ_5	γ_6	γ_7	γ_8	γ_9	γ_{10}
−0.0236	0.2414	0.1762	0.1151	0.0186	0.0340	0.3162	−0.0201
ΔH	ΔL	ΔA	ΔP	ΔC	ΔQ	ΔR	ΔM
γ_{11}	γ_{12}	γ_{13}	γ_{14}	γ_{15}	γ_{16}	γ_{17}	γ_{18}
0.0465	0.2349	0.0953	−0.0151	0.1221	−0.0303	0.0988	−0.0245
ΔT	ΔS	Δgreen:J	Δorange:J	Δpurple:J	Δred:J	Δyellow:J	Δgreen:U
γ_{19}	γ_{20}	$\delta_{1,1}$	$\delta_{2,1}$	$\delta_{3,1}$	$\delta_{4,1}$	$\delta_{5,1}$	$\delta_{1,2}$
0.1824	−0.0048	0.1827	0.0758	-0.0299	−0.0336	0.2052	−0.0082
Δorange:U	Δpurple:U	Δred:U	Δyellow:U	Δgreen:O	Δorange:O	Δpurple:O	Δred:O
$\delta_{2,2}$	$\delta_{3,2}$	$\delta_{4,2}$	$\delta_{5,2}$	$\delta_{1,3}$	$\delta_{2,3}$	$\delta_{3,3}$	$\delta_{4,3}$

0.1243	0.0413	−0.0219	0.0477	0.0139	0.0633	0.4694	0.0158
Δyellow:O	Δgreen:N	Δorange:N	Δpurple:N	Δred:N	Δyellow:N	Δgreen:D	Δorange:D
$\delta_{5,3}$	$\delta_{1,4}$	$\delta_{2,4}$	$\delta_{3,4}$	$\delta_{4,4}$	$\delta_{5,4}$	$\delta_{1,5}$	$\delta_{2,5}$
0.0081	−0.0055	0.0248	−0.0454	−0.0274	0.0591	0.2258	−0.0364
Δpurple:D	Δred:D	Δyellow:D	Δgreen:B	Δorange:B	Δpurple:B	Δred:B	Δyellow:B
$\delta_{3,5}$	$\delta_{4,5}$	$\delta_{5,5}$	$\delta_{1,6}$	$\delta_{2,6}$	$\delta_{3,6}$	$\delta_{4,6}$	$\delta_{5,6}$
−0.0530	−0.0035	0.2085	0.0156	0.0073	0.0262	0.0300	0.0524
Δgreen:G	Δorange:G	Δpurple:G	Δred:G	Δyellow:G	Δgreen:F	Δorange:F	Δpurple:F
$\delta_{1,7}$	$\delta_{2,7}$	$\delta_{3,7}$	$\delta_{4,7}$	$\delta_{5,7}$	$\delta_{1,8}$	$\delta_{2,8}$	$\delta_{3,8}$
−0.0081	0.0532	−0.0334	−0.0331	0.0223	0.0475	0.1130	0.0654
Δred:F	Δyellow:F	Δgreen:E	Δorange:E	Δpurple:E	Δred:E	Δyellow:E	Δgreen:I
$\delta_{4,8}$	$\delta_{5,8}$	$\delta_{1,9}$	$\delta_{2,9}$	$\delta_{3,9}$	$\delta_{4,9}$	$\delta_{5,9}$	$\delta_{1,10}$
0.0157	0.0732	0.0108	0.1009	−0.0266	−0.0230	0.0330	0.0010
Δorange:I	Δpurple:I	Δred:I	Δyellow:I	Δgreen:H	Δorange:H	Δpurple:H	Δred:H
$\delta_{2,10}$	$\delta_{3,10}$	$\delta_{4,10}$	$\delta_{5,10}$	$\delta_{1,11}$	$\delta_{2,11}$	$\delta_{3,11}$	$\delta_{4,11}$
0.0283	0.0109	−0.0324	0.0145	0.2966	0.2053	0.1530	0.1449
Δyellow:H	Δgreen:L	Δorange:L	Δpurple:L	Δred:L	Δyellow:L	Δgreen:A	Δorange:A
$\delta_{5,11}$	$\delta_{1,12}$	$\delta_{2,12}$	$\delta_{3,12}$	$\delta_{4,12}$	$\delta_{5,12}$	$\delta_{1,13}$	$\delta_{2,13}$
0.0655	0.0664	−0.0124	−0.0446	−0.0373	0.0513	0.0335	0.0089
Δpurple:A	Δred:A	Δyellow:A	Δgreen:P	Δorange:P	Δpurple:P	Δred:P	Δyellow:P
$\delta_{3,13}$	$\delta_{4,13}$	$\delta_{5,13}$	$\delta_{1,14}$	$\delta_{2,14}$	$\delta_{3,14}$	$\delta_{4,14}$	$\delta_{5,14}$
0.1640	0.0916	0.1321	0.2461	0.0003	0.3432	0.1339	0.3503
Δgreen:C	Δorange:C	Δpurple:C	Δred:C	Δyellow:C	Δgreen:Q	Δorange:Q	Δpurple:Q
$\delta_{1,15}$	$\delta_{2,15}$	$\delta_{3,15}$	$\delta_{4,15}$	$\delta_{5,15}$	$\delta_{1,16}$	$\delta_{2,16}$	$\delta_{3,16}$
0.0511	0.1734	0.1181	0.1088	0.0275	0.0927	0.5038	0.4851
Δred:Q	Δyellow:Q	Δgreen:R	Δorange:R	Δpurple:R	Δred:R	Δyellow:R	Δgreen:M
$\delta_{4,16}$	$\delta_{5,16}$	$\delta_{1,17}$	$\delta_{2,17}$	$\delta_{3,17}$	$\delta_{4,17}$	$\delta_{5,17}$	$\delta_{1,18}$
−0.0018	−0.0088	0.0183	0.0265	0.0246	0.1470	0.0906	0.0454
Δorange:M	Δpurple:M	Δred:M	Δyellow:M	Δgreen:T	Δorange:T	Δpurple:T	Δred:T
$\delta_{2,18}$	$\delta_{3,18}$	$\delta_{4,18}$	$\delta_{5,18}$	$\delta_{1,19}$	$\delta_{2,19}$	$\delta_{3,19}$	$\delta_{4,19}$
0.2919	−0.0034	0.0584	0.1556	−0.0172	0.0164	−0.0052	−0.0257
Δyellow:T	Δgreen:S	Δorange:S	Δpurple:S	Δred:S	Δyellow:S		
$\delta_{5,19}$	$\delta_{1,20}$	$\delta_{2,20}$	$\delta_{3,20}$	$\delta_{4,20}$	$\delta_{5,20}$		
−0.0095	−0.0071	−0.0007	−0.0173	0.1525	−0.0019		

Figure 14 shows box plots of the fitted values, first grouped by data sets and then grouped by algorithms. Again, in comparison with the box plot of the raw performance measures, Figure 7, no serious differences are visible. The residual plot, Figure 15, shows that all mean values are near zero and all deviations are symmetric, except algorithm green in some cases, e.g., data set J.

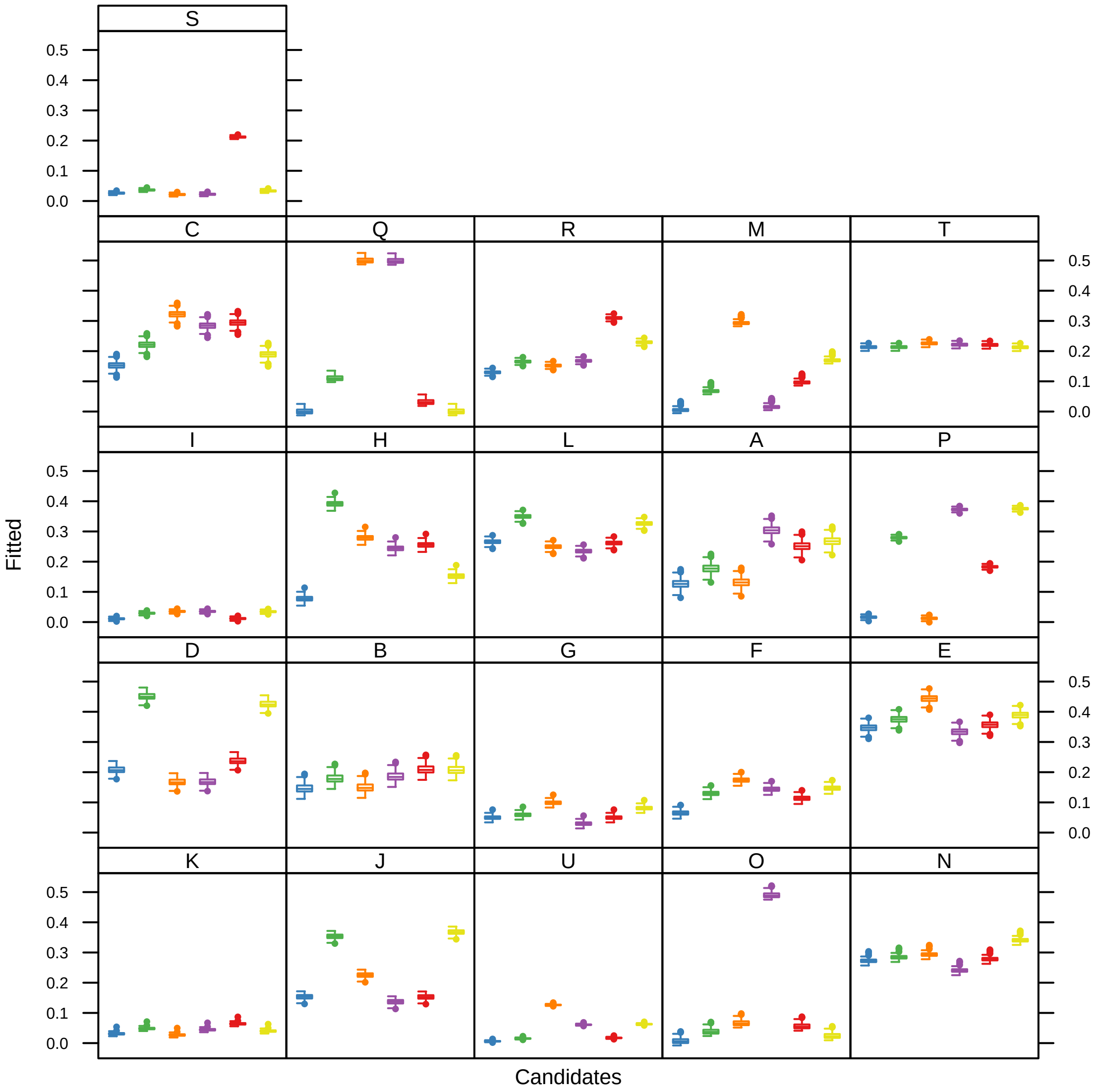


Figure 14: Mixed effects model of the exemplar benchmark survey: box plot of fitted values, grouped by data sets.

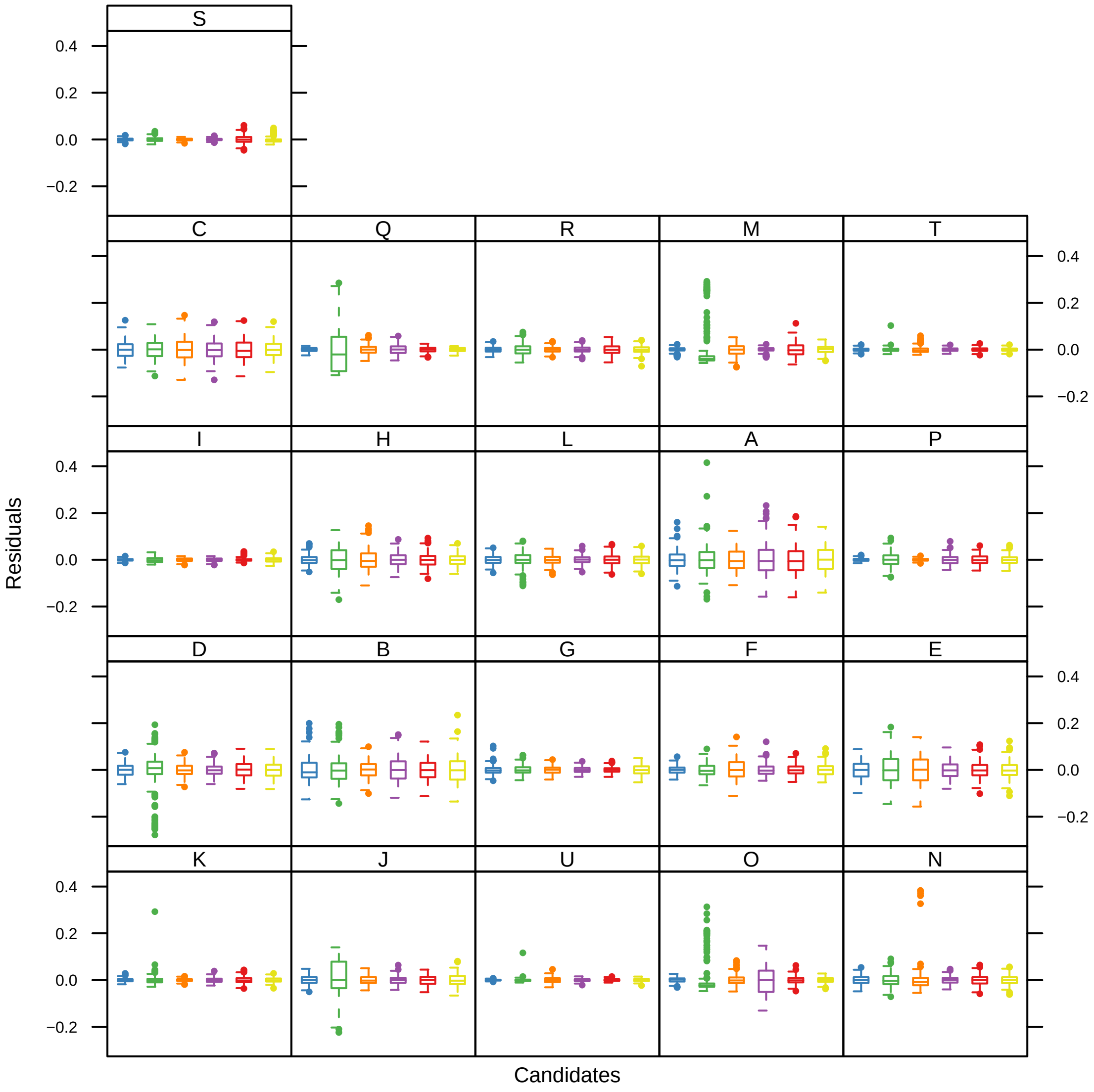


Figure 15: Mixed effects model of the exemplar benchmark survey: box plot of the residuals, grouped by data sets.

B. Components of the exemplar benchmark experiment

B.1. Algorithms

The candidate algorithms are:

linear discriminant analysis: encoded as 🟧, `orange`; available through the function `lda` in package `MASS`.

naive bayes classifier: encoded as 🟨, `yellow`; available through the function `naiveBayes` in package `e1071`.

k-nearest neighbour classifier: encoded as 🟪, `purple`; available through the function `knn` in package `class`. The hyperparameter k (the number of neighours) is determined with cross-validation between 1 and $\sqrt{n}$, n the number of observations.

classification trees: encoded as 🟥, `red`; available through the function `rpart` in package `rpart`. The fulled tree is pruned according to the 1-SE rule (e.g., Venables and Ripley, 2002; Hastie et al., 2001).

support vector machines: encoded as 🟦, `blue`; available through the function `svm` in package `e1071`. We use the *C-classification* machine, which has two hyperparameters γ (the cost of constraints violation) and c (the kernel parameter). Following Meyer et al. (2003) the best choices are determined with a grid search over the two-dimensional parameter space (γ, c), γ ranges from 2^{-5} to 2^{12} and c from 2^{-10} to 2^{5}.

neural networks: encoded as 🟩, `green`; available through the function `nnet` in package `nnet`. The hyperparameter is the number of hidden units. The best value is searched with cross-validation between 1 and $\log(n)$, n the number of observations (following Meyer et al., 2003).

B.2. Data sets

The benchmark survey is made up of 21 binary classification problems originated from the UCI Machine Learning repository (Asuncion and Newman, 2007):

Problem		#Attributes		#Samples		Class
		nominal	continuous	complete	incomplete	distribution (%)
promotergene	`A`	57		106		50.00/50.00
hepatitis	`B`	13	6	80	75	20.65/79.35
Sonar	`C`		60	208		53.37/46.63
Heart1	`D`	8	5	296	7	54.46/45.54
liver	`E`		6	345		42.03/57.97
Ionosphere	`F`	1	32	351		35.90/64.10
HouseVotes84	`G`	16		232	203	61.38/38.62
musk	`H`		166	476		56.51/43.49
monks3	`I`	6		554		48.01/51.99
Cards	`J`	9	6	653	37	44.49/55.51
BreastCancer	`K`	9		683	16	65.52/34.48
PimaIndiansDiabetes	`L`		8	768		65.10/34.90
tictactoe	`M`	9		958		34.66/65.34
credit	`N`		24	1000		70.00/30.00
Circle (*)	`O`		2	1200		50.67/49.33

ringnorm (*)	`P`		20	1200	50.00/50.00
Spirals (*)	`Q`		2	1200	50.00/50.00
threenorm (*)	`R`		20	1200	50.00/50.00
twonorm (*)	`S`		20	1200	50.00/50.00
titanic	`T`	3		2201	67.70/32.30
chess	`U`	36		3196	47.78/52.22

Table 4: The 21 problems used for illustration. Problems marked with (*) are artificially created. The list ist sorted by the number of samples. The letters are codes for plots which deal with these problems.

References

A. Asuncion and D.J. Newman. UCI machine learning repository, 2007. URL `http://www.ics.uci.edu/ mlearn/MLRepository.html`.

Richard A. Becker, William S. Cleveland, and Ming-Jen Shyu. The visual design and control of trellis display. *Journal of Computational and Graphical Statistics*, 5(2):123–155, 1996.

Janez Demsar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

Manuel J. A. Eugster and Friedrich Leisch. Bench plot and mixed effects models: First steps toward a comprehensiv benchmark analysis toolbox. Technical Report 26, Institut für Statistik, Ludwig-Maximilians-Universität München, Germany, 2008.

Emden R. Gansner and Stephen C. North. An open graph visualization system and its applications to software engineering. *Software — Practice and Experience*, 30(11):1203–1233, 2000.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer-Verlag, 2001.

Myles Hollander and Douglas A. Wolfe. *Nonparametric Statistical Methods.* John Wiley & Sons, 2nd edition, 1999.

Kurt Hornik and David Meyer. Deriving consensus rankings from benchmarking experiments. In R. Decker and H.-J. Lenz, editors, *Advances in Data Analysis (Proceedings of the 30th Annual Conference of the Gesellschaft für Klassifikation e.V., Freie Universität Berlin, March 8–10, 2006*, Studies in Classification, Data Analysis, and Knowledge Organization, pages 163–170. Springer-Verlag, 2007.

Torsten Hothorn, Friedrich Leisch, Achim Zeileis, and Kurt Hornik. The design and analysis of benchmark experiments. *Journal of Computational and Graphical Statistics*, 14(3):675–699, 2005.

Torsten Hothorn, Kurt Hornik, Mark A. van de Wiel, and Achim Zeileis. A Lego system for conditional inference. *The American Statistician*, 60(3), 2006.

Torsten Hothorn, Frank Bretz, and Peter Westfall. Simultaneous inference in general parametric models. *Biometrical Journal*, 50(3), 2008.

Donald E. Knuth. *The Art of Computer Programming: Fundamental Algorithms.* Addison-Wesley, third edition, 1997. ISBN 0-201-89683-4.

David Meyer, Friedrich Leisch, and Kurt Hornik. The support vector machine under test. *Neurocomputing*, 55:169–186, September 2003.

José C. Pinheiro and Douglas M. Bates. *Mixed-Effects Models in S and S-PLUS.* Springer-Verlag, 2000. ISBN 0-387-98957-9.

R Development Core Team. *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria, 2008. URL `http://www.R-project.org`. ISBN 3-900051-07-0.

William Venables and Brian Ripley. *Modern Applied Statistics with S.* Springer-Verlag, fourth edition, 2002.

Olcay Taner Yildiz and Ethem Alpaydin. Ordering and finding the best of $k > 2$ supervised learning algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(3): 392–402, 2006.