Supplementary Materials to the Paper:

Evaluating machine learning models in non-standard settings: An overview and new findings

Roman Hornung*[,1,2], Malte Nalenz [3], Lennart Schneider [3,2], Andreas Bender [3,2], Ludwig Bothmann [3,2], Bernd Bischl [3,2], Thomas Augustin [3], Anne-Laure Boulesteix [1,2]

[1] Institute for Medical Information Processing, Biometry and Epidemiology, LMU Munich, Munich, Germany

[2] Munich Center for Machine Learning (MCML), Munich, Germany

[3] Department of Statistics, LMU Munich, Munich, Germany

*To whom correspondence should be addressed: hornung@ibe.med.uni-muenchen.de

Contents

A Simulation study comparing GE estimation with and without respecting the clustering structure: Full design and extensive results

A.1 Simulation design

As described in the main paper, the data were simulated according to the following model:

$$y^{(mi)} = \sum_{l=1}^{5} \beta_l x_l^{(mi)} + b_{m1} + b_{m2} x_1^{(mi)} + \epsilon^{(mi)},$$

where $\epsilon^{(mi)} \sim \mathcal{N}(0, \sigma^2)$, $b_{m1} \sim \mathcal{N}(0, \sigma_1^2)$, and $b_{m2} \sim \mathcal{N}(0, \sigma_2^2)$. Moreover, we set $(\beta_1, \beta_2, \beta_3, \beta_4, \beta_5)^T = (1, 1, -1, 0, 0)^T$.

We considered the following setting parameters:

- Number of clusters: $N \in \{10, 50\}$
- Number of observations per cluster: $n_m \in \{5, 25\}$
- Cluster-specific means (yes vs. no): $\sigma_1^2 \in \{1, 0\}$
- Cluster-specific influence of x_1 (yes vs. no): $\sigma_2^2 \in \{1, 0\}$
- Signal strength (strong vs. weak): $\sigma^2 \in \{0.25, 1\}$
- Feature constant within clusters: none, x_1, x_2

By considering every possible combination out of these setting parameter values we investigated 96 simulation settings in total ($2 \times 2 \times 2 \times 2 \times 2 \times 3$). For each setting we simulated 100 data sets.

A.2 Simulation results

The results obtained using linear models and random forests are shown in Supplementary Figures S1–S3 and S4–S6, respectively. To avoid redundancy, we will primarily describe the results obtained using linear models and highlight differences to the results obtained using random forests.

Supplementary Figures S1 and S4 show the results obtained for the settings without cluster-constant feature values. As expected, for settings with cluster-specific means and effects ($\sigma_1^2 = \sigma_2^2 = 0$) there were no differences

between the results obtained for standard and grouped CV (apart from small random fluctuations). For cluster-specific means and effects, the MSE estimates were slightly larger for grouped CV, but only for $N = 10$, the smaller of the two considered numbers of clusters. For smaller numbers of clusters the training data sets consisted of fewer clusters, which is why the learned model was more strongly influenced by each individual cluster. This likely explains why we only observed a difference in the results for the smaller considered number of clusters. There were no notable differences in the results between the settings that consider cluster-specific means ($\sigma_1^2 = 1$), cluster-specific effects ($\sigma_2^2 = 1$), and both, cluster-specific means and effects ($\sigma_1^2 = \sigma_2^2 = 1$). The number of observations per cluster n_m and the strength of the signal (controlled by σ^2) had no notable influence on the results.

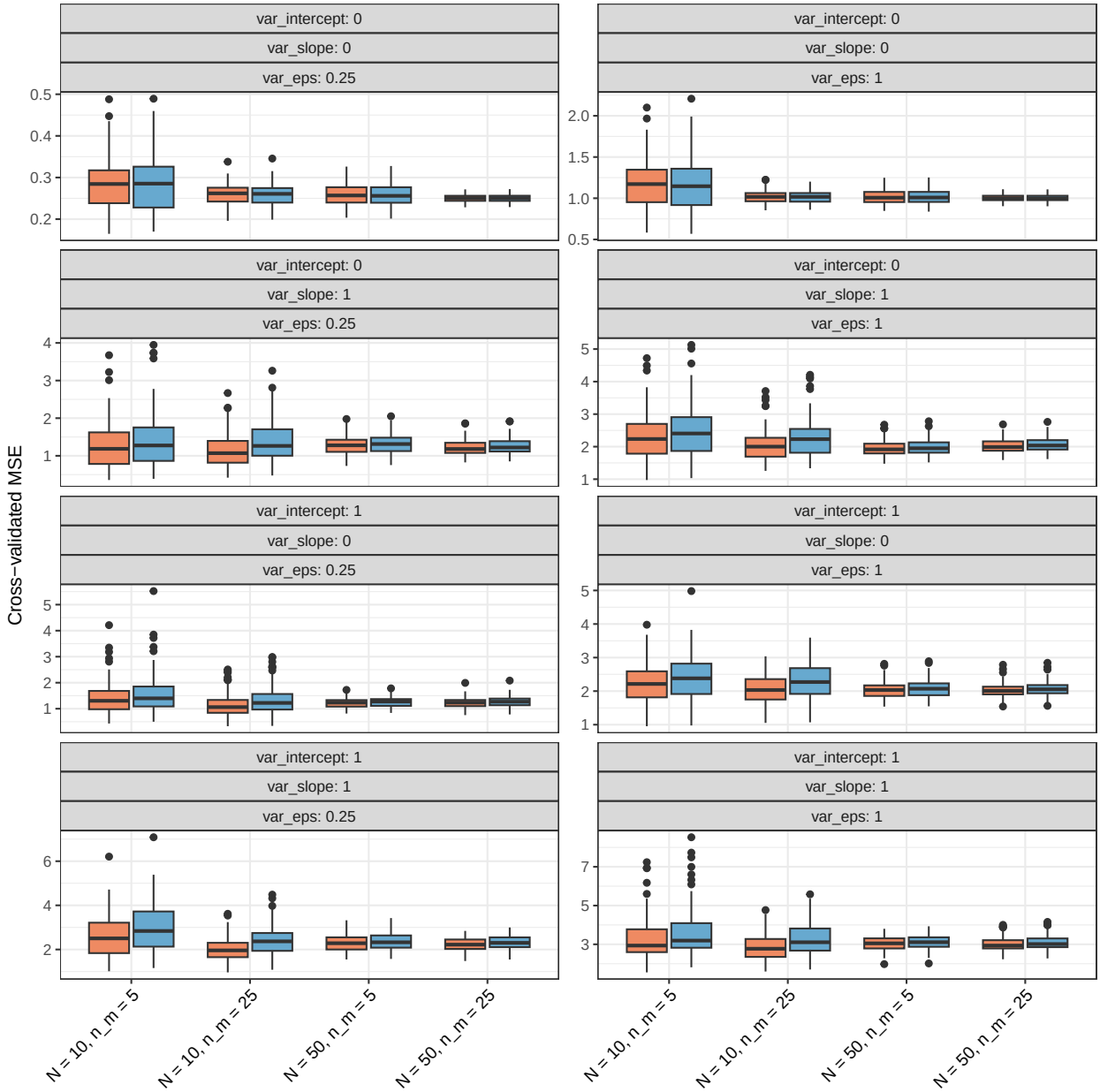

Fig. S1: Simulation on clustered data: cross-validated MSE values of the linear models for each setting without cluster-constant feature values. The orange and the blue boxplots indicate the results obtained for standard and grouped CV, respectively.

For the linear models, the results obtained for the settings in which x_1 or x_2 were constant within clusters are shown in Supplementary Figures S2 and S3, respectively. The same results for the random forests are shown in Supplementary Figures S5 and S6. For both learners, we observed that the differences between the two CV variants was stronger than for the settings without cluster-constant feature values. Beyond this commonality, there were,

however, also quite strong differences in the results obtained for the two learners. In the case of the random forests, the MSE estimates were considerably larger for grouped than for standard CV for almost all settings. Regarding the linear models, the differences between the two CV variants were less pronounced and only seen for a subset of the settings. Notable differences were only observed for settings with cluster-specific means and/or effects, and only for $N = 10$.

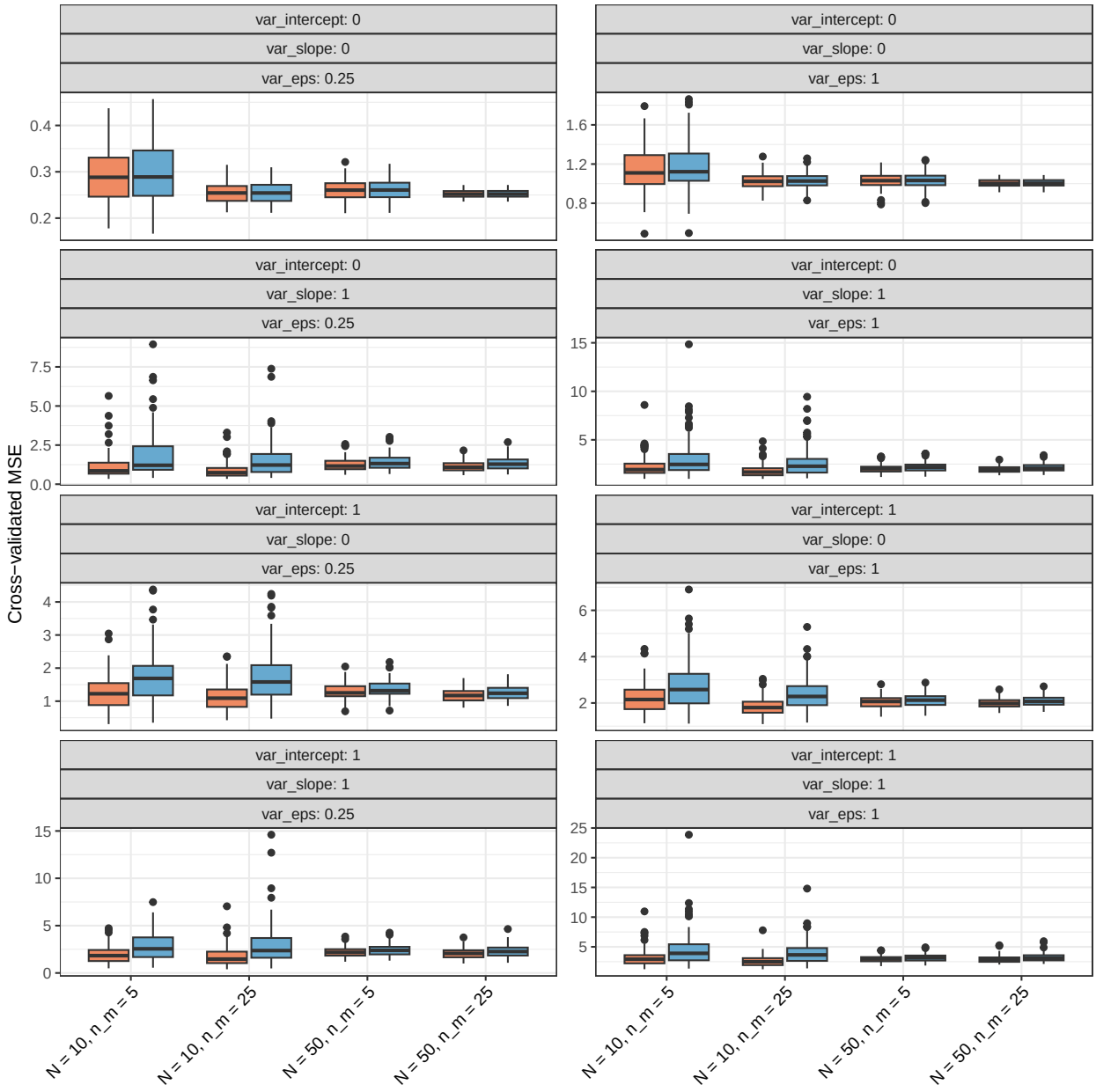

Fig. S2: Simulation on clustered data: cross-validated MSE values of the linear models for each setting for which the x_1 values were constant within clusters. The orange and the blue boxplots indicate the results obtained for standard and grouped CV, respectively.

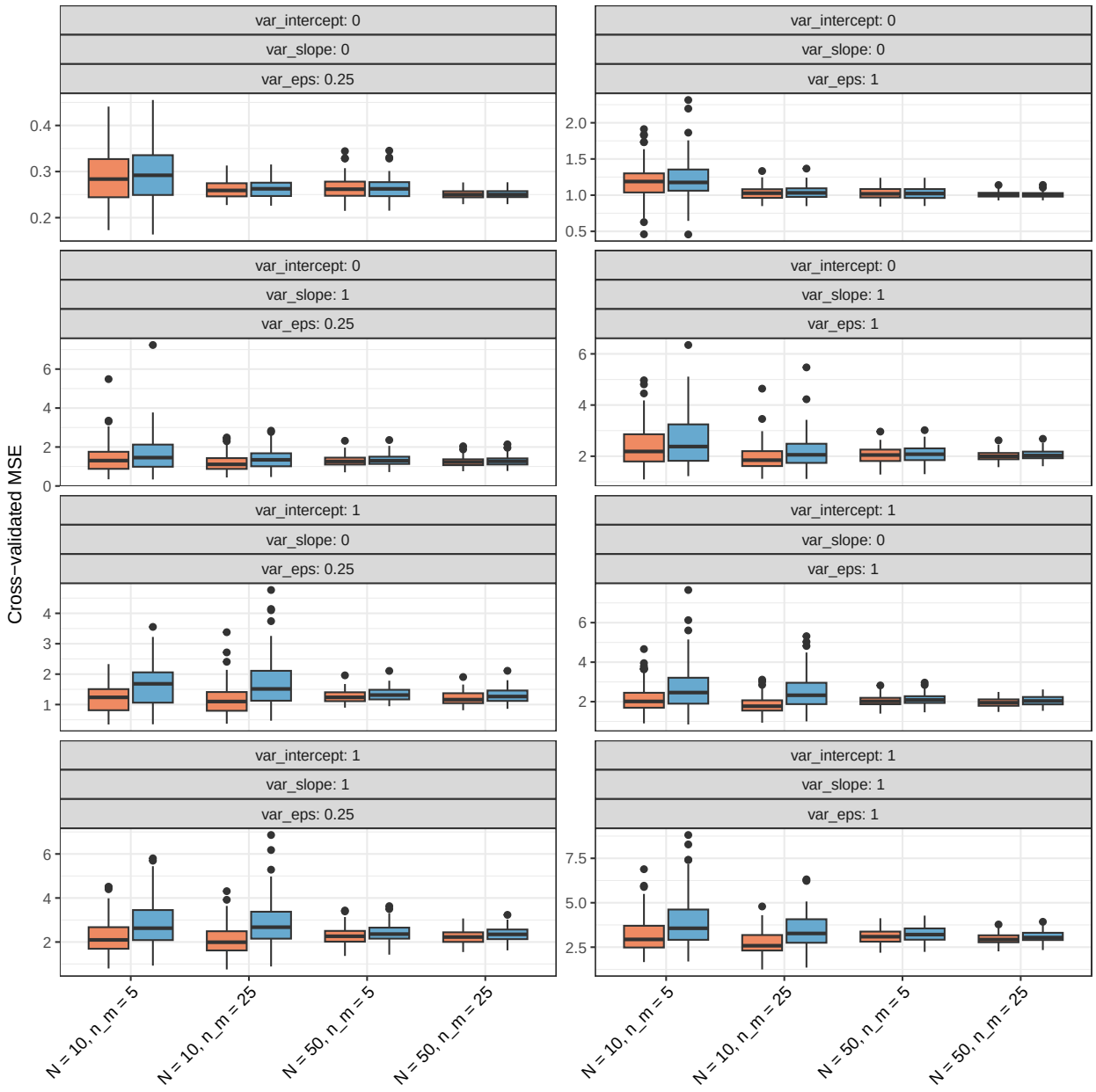

Fig. S3: Simulation on clustered data: cross-validated MSE values of the linear models for each setting for which the x_2 values were constant within clusters. The orange and the blue boxplots indicate the results obtained for standard and grouped CV, respectively.

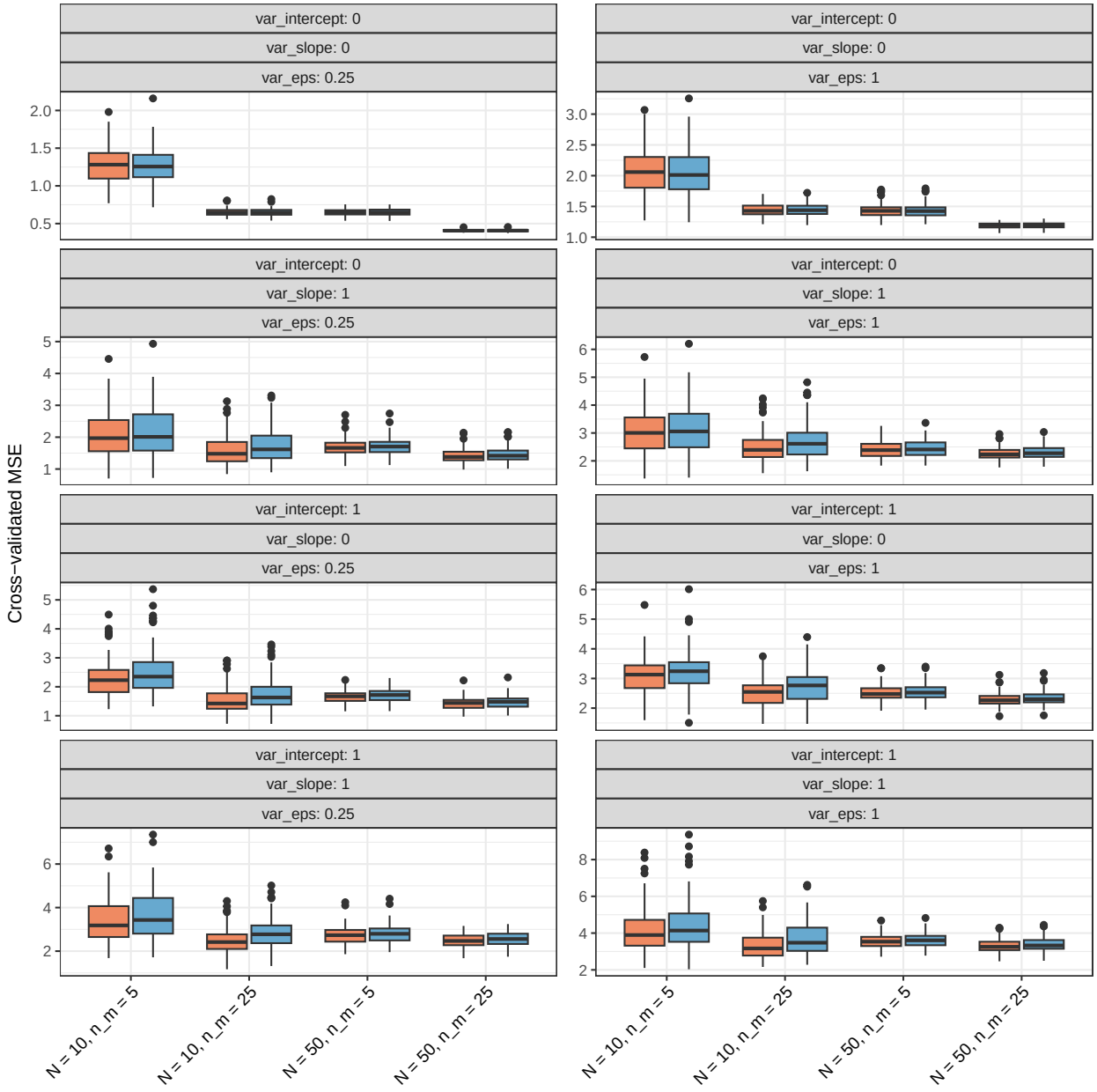

Fig. S4: Simulation on clustered data: cross-validated MSE values of the random forests without cluster-constant feature values. The orange and the blue boxplots indicate the results obtained for standard and grouped CV, respectively.

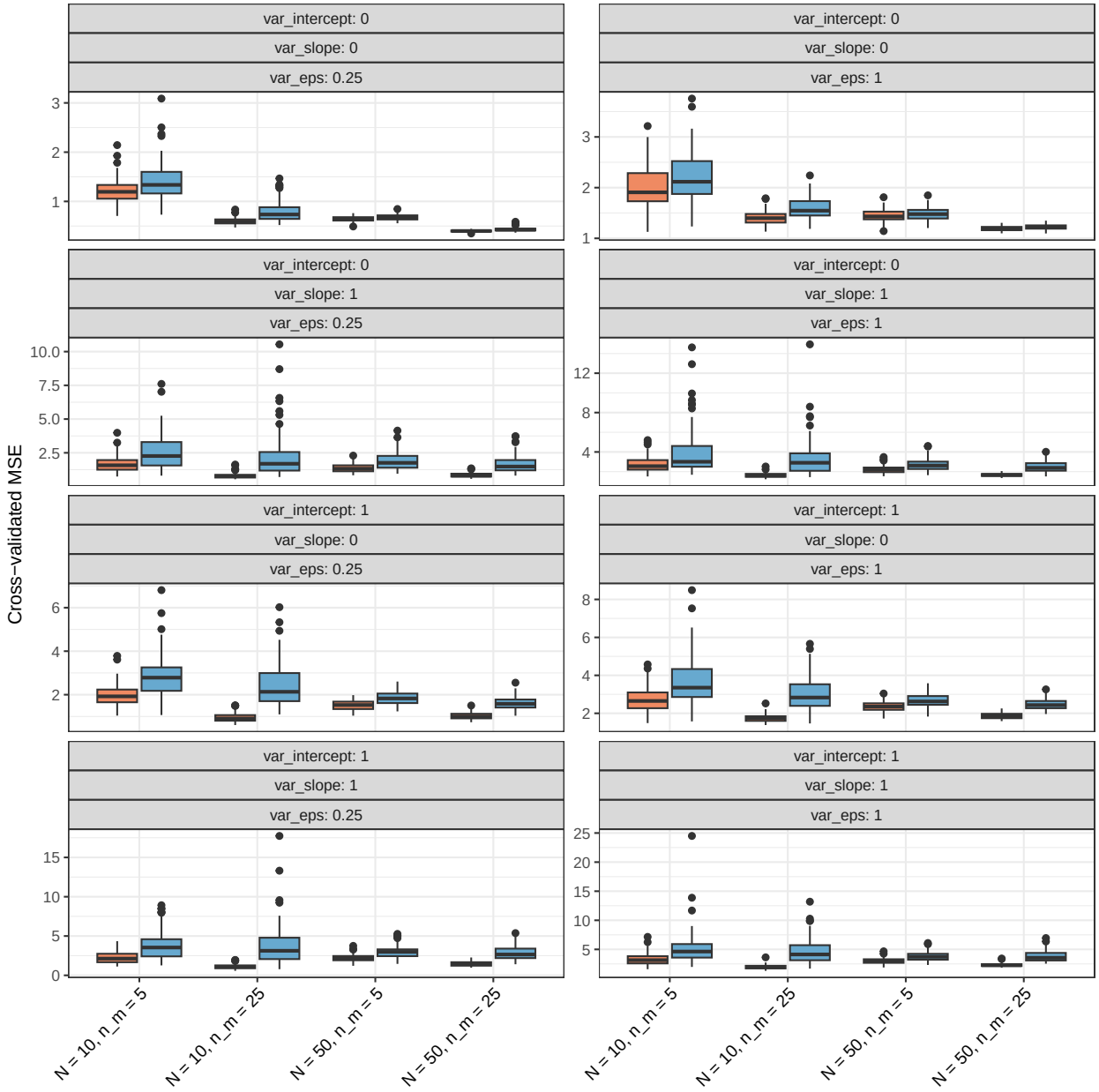

Fig. S5: Simulation on clustered data: cross-validated MSE values of the random forests for which the x_1 values were constant within clusters. The orange and the blue boxplots indicate the results obtained for standard and grouped CV, respectively.

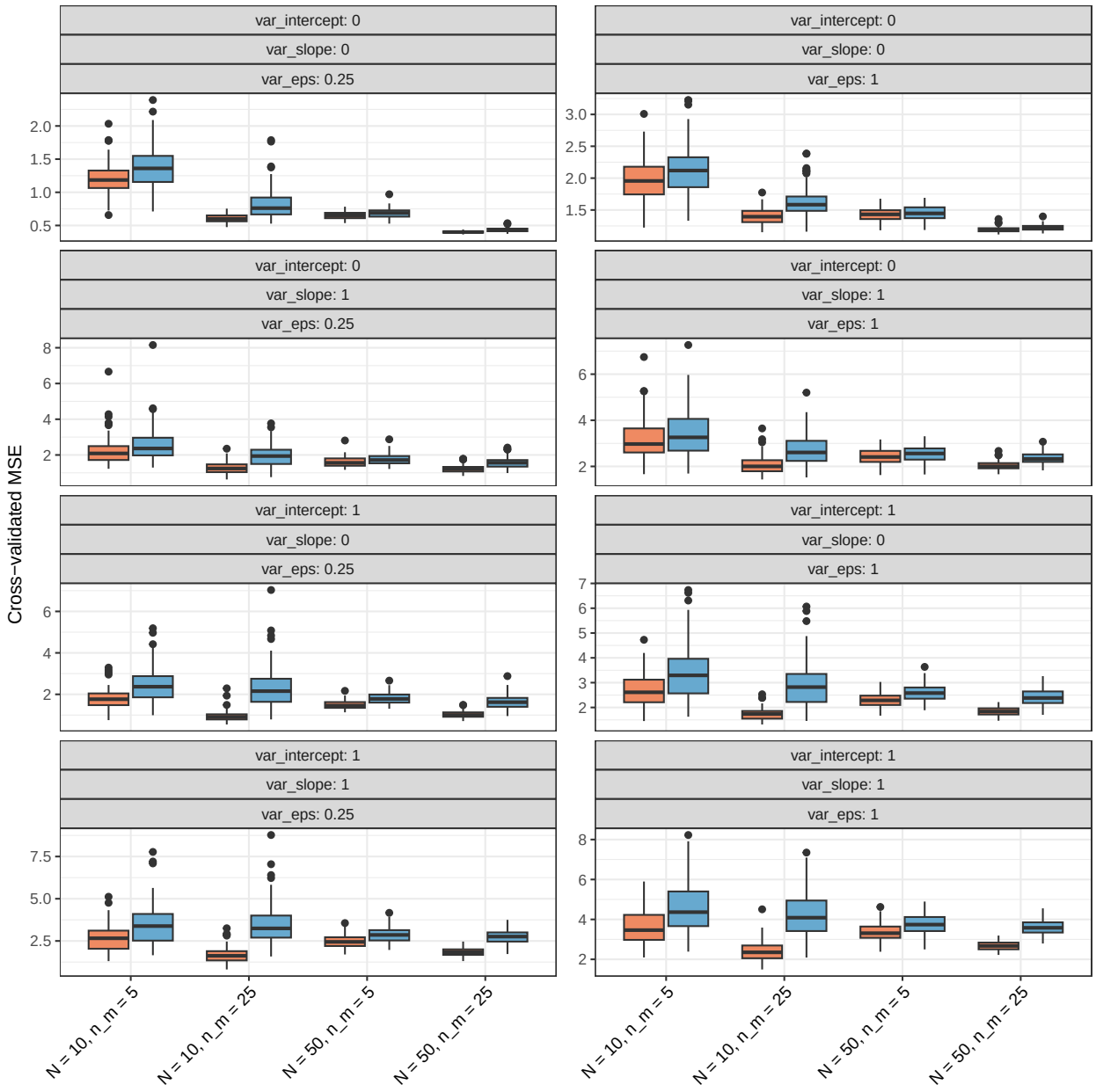

Fig. S6: Simulation on clustered data: cross-validated MSE values of the random forests for which the x_2 values were constant within clusters. The orange and the blue boxplots indicate the results obtained for standard and grouped CV, respectively.

B Simulation study comparing different approaches to GE estimation under concept drift: Full design and extensive results

B.1 Simulation design

We simulated data according to the following model:

$$y^{(t)} = \beta_0(t) + \sum_{l=1}^{5} \beta_l x_l^{(t)} + \epsilon^{(t)}, \quad t \in [0,1], \tag{1}$$

where $x_{l_1}^{(t)} \sim \mathcal{N}(\mu_{l_1}(t), 1)$ for $l_1 \in \{1,2,3\}$, $x_{l_2}^{(t)} \sim \mathcal{N}(0,1)$ for $l_2 \in \{4,5\}$, and $\epsilon^{(t)} \sim \mathcal{N}(0, \sigma^2(t))$. We set $(\beta_1, \beta_2, \beta_3, \beta_4, \beta_5)^T = (2, -1, 2, 0, 0)^T$. The time-dependent label mean $\beta_0(t)$ and variance $\sigma^2(t)$ of $y^{(t)}$ as well as the time-dependent features means $\mu_1(t)$, $\mu_2(t)$, and $\mu_3(t)$ depended linearly on t. This corresponds to a constant degree of change in the distributions of $y^{(t)}$ and $x_{l_1}^{(t)}$ ($l_1 \in \{1,2,3\}$).

For the strength of the drifts in the distributions of $y^{(t)}$ and $x_{l_1}^{(t)}$ ($l_1 \in \{1,2,3\}$), that is, the label shift and the feature shift, respectively, we considered four levels: "none", "weak", "medium-strong", and "strong". These are characterized as follows:

- "None": for all $t \in [0,1]$: $\beta_0(t) = 0$, $\sigma^2(t) = 1$, $\mu_{l_1}(t) = 0$ for $l_1 \in \{1,2,3\}$
- "Weak": at the beginning, that is, for $t = 0$: $\beta_0(0) = 0$, $\sigma^2(0) = 1$, $\mu_{l_1}(0) = 0$ for $l_1 \in \{1,2,3\}$; at the end, that is, for $t = 1$: $\beta_0(1) = 2$, $\sigma^2(1) = 2$, $\mu_1(1) = 2$, $\mu_2(1) = 1$, $\mu_3(1) = -2$
- "Medium-strong": at the beginning, that is, for $t = 0$: $\beta_0(0) = 0$, $\sigma^2(0) = 1$, $\mu_{l_1}(0) = 0$ for $l_1 \in \{1,2,3\}$; at the end, that is, for $t = 1$: $\beta_0(1) = 4$, $\sigma^2(1) = 4$, $\mu_1(1) = 4$, $\mu_2(1) = 2$, $\mu_3(1) = -4$
- "Strong": at the beginning, that is, for $t = 0$: $\beta_0(0) = 0$, $\sigma^2(0) = 1$, $\mu_{l_1}(0) = 0$ for $l_1 \in \{1,2,3\}$; at the end, that is, for $t = 1$: $\beta_0(1) = 8$, $\sigma^2(1) = 8$, $\mu_1(1) = 8$, $\mu_2(1) = 4$, $\mu_3(1) = -8$

We divided the time span $t \in [0,1]$ into ten adjacent intervals of equal length. The first eight of these intervals, hereafter referred to as "seasons", represented the training data available to the analyst. Supplementary Figures S7 and S8 visualize the strengths of the feature and label drifts in the 10 seasons. After each of these seasons, the analyst may re-train and/or re-validate models trained with these data. The last two seasons represented the future for which predictions should be made using the final models trained on the first eight seasons, that is, on the training data.

Each of the n_{train} training samples, $n_{\text{train}} \in \{100, 500, 1000, 3000\}$, was simulated at a different time t_i, $i \in \{1, \ldots, n\}$, where these times were placed equidistantly on the interval starting at zero and ending at the end of the eighth season (i.e., $t_1 = 0, t_n = 0.8$).

The true prediction error (i.e., the GE) of the forecast was approximated at five different points in time: 1) the end of the eighth season, 2) the middle of the ninth season, 3) the end of the ninth season, 4) the middle of the tenth season, and 5) the end of the tenth season. At each of these time points, we generated a large test data set of $2 \cdot 10^5$ observations (again using model (1)). To approximate the GEs at these time points, we trained the model on the entire training data set and evaluated it on the large test data sets.

We compared the following methods for estimating the GE:

- *k-fold CV*: We used 8-fold CV, repeated ten times. The reason for choosing $k = 8$ in the CV was to obtain the same fold sizes as in the time-series CV (see below). This procedure will be referred to as `CV`.
- *Time-series CV/prequential evaluation*: First, the data were split into k folds of (approximately) equal size according to time. That is, the n/k observations that occurred first were placed in the first fold, the next n/k observations were placed in the second fold, and so on. Second, the model was trained on the first fold and evaluated on the second fold, then trained on the first two folds and evaluated on the third fold, then trained on the first three folds and evaluated on the fourth fold, and so on. Finally, the $k - 1$ GE estimates were averaged to obtain the final GE estimate. Choosing $k = 8$ had the effect that the folds coincided with

the eight seasons in the training data. Since in each iteration of this procedure we are making predictions for the next season after the respective training folds, this measures the GE to expect one season ahead after training. However, as concept drift progresses, the GE will be higher further into the future if no retraining is performed. To estimate the GE in the season after the next following training, we included a version of the procedure in which we did not use the next but the next but one folds as validation folds. This and the original version of the time-series CV will be referred to as `CV2s` and `CV1s`, respectively.

- *Out-of-sample validation*: This is a commonly used technique in time series forecasting (Bergmeir et al., 2018). The data are split once into training and validation data, where all observations that occurred before a certain point in time are placed in the training data and the rest in the validation data. We used the first seven seasons for training and the eighth season for validation. It is also common to include a buffer zone between the training and validation data when the interest is in estimating the GE to be expected further in the future. We also included a variant where the first six seasons were used for training and the eighth season for validation. The goal of this procedure was to estimate the GE to be expected in the season after next. We will refer to this and the original variant as `OOS2s` and `OOS1s`, respectively. Out-of-sample validation has the advantage over time-series CV that the validation data used are more similar in distribution to the future data, which may lead to more realistic GE estimates.

We considered every possible combination of the four training set sizes, the four levels of feature shift, and the four levels of label shift, resulting in a total of 64 simulation settings ($4 \times 4 \times 4$). We simulated 100 data sets for each setting.

B.2 Results

Supplementary Figures S9–S12 and Supplementary Figures S13–S16 show the estimated and (approximated) true MSE values obtained for linear models and random forests, respectively.

`OOS1s` is the only method that provided nearly unbiased GE estimates across all simulation settings. However, in many cases, these GE estimates were unbiased only at the end of the training period, because the true GE often increased quickly thereafter. Thus, `OOS1s` can only be used to obtain realistic GE estimates for the period immediately following training, as the corresponding GE estimates are likely to be too optimistic for later time points. A drawback of `OOS1s` is, however, that the variance of the GE estimates is quite large, especially for small training set sizes. Also, in some cases the GE estimates were slightly pessimistic, which should, however, not be a problem in most situations, since slightly pessimistic GE estimates are much generally less dangerous than overoptimistic ones.

`CV` provided strongly overoptimistic performance estimates in most cases. More specifically, for the random forests, it only provided (almost) unbiased GE estimates when there was no concept drift at all. For the linear models, it provided unbiased GE estimates for settings with pure feature shift but no label shift. The latter observation can be explained by the fact that the data were simulated using a linear model. When there is pure feature drift and the data are simulated with a linear model, the GE does not (notably) increase after training because the linear models fitted to the training data can extrapolate to regions of the feature space that are not contained in the training data. In contrast, random forests are not suitable for making predictions for regions outside the feature space observed in the training data. This is because random forests make predictions in these regions that do not depend on the distance of the corresponding observations from those in the training data.

The GE estimates obtained with `CV1s` were only slightly less overoptimistic than those obtained with `CV`. In particular, for moderate to strong label shift, the GE estimates obtained with `CV1s` tended to be much smaller than the GEs at the end of the training period. Moreover, in contrast to `CV`, the GE estimates were notably pessimistic in settings with no or weak concept drift. This can probably be explained by the fact that the training set sizes in `CV1s` were generally much smaller than the entire training data set (on average only $n_{\text{train}}/2$ observations).

`CV2s` behaved very similarly to `CV1s`, in particular it produced strongly overoptimistic estimates for the GE to be expected in the tenth season in most cases.

In many cases, `OOS2s` produced reasonable estimates of the GE to be expected at the beginning or middle of the tenth season. Similar to `OOS1s`, the variance of `OOS2s` was quite high. This was especially true for smaller training set sizes. More worryingly, unlike `OOS1s`, there were also many settings with strong label drift for which `OOS2s` was associated with overoptimism, although the latter was not very strong. There were some differences between the results obtained with linear models and random forests on this point.

In general, however, it is important not to over-interpret the details of the results obtained, as they likely depended on the specific simulation design chosen. For example, for settings with moderate and strong label shifts, the true and estimated errors became smaller for stronger feature shifts, which seems surprising. However, it can be explained by considering the specific simulation design used. For settings with label shift, the intercept $\beta_0(t)$ increased with t. Moreover, for settings with feature shift, $\beta_1 x_1^{(t)} + \beta_2 x_2^{(t)} + \beta_3 x_3^{(t)}$ had a slightly decreasing trend with t because the positive and negative components $\beta_1 x_1^{(t)}$ and $\beta_3 x_3^{(t)}$, respectively, canceled each other out, and $\beta_2 x_2^{(t)}$ had a slightly decreasing trend with t. From the combination that $\beta_0(t)$ was increasing and $\beta_1 x_1^{(t)} + \beta_2 x_2^{(t)} + \beta_3 x_3^{(t)}$ was decreasing, it follows that the linear predictor $\beta_0(t) + \beta_1 x_1^{(t)} + \beta_2 x_2^{(t)} + \beta_3 x_3^{(t)}$ had a smaller variance than in the settings without feature shift, because in the latter settings $\beta_1 x_1^{(t)} + \beta_2 x_2^{(t)} + \beta_3 x_3^{(t)}$ had no decreasing trend. From this we can see that in the subset of settings with label shift, the variance of $y^{(t)}$ was smaller for settings with feature drift. This finally explains why the true and estimated prediction errors were smaller in these settings. When there was no label shift, the variance of $\beta_0(t) + \beta_1 x_1^{(t)} + \beta_2 x_2^{(t)} + \beta_3 x_3^{(t)}$ was not smaller for settings with than without feature shift, and thus the prediction errors were not smaller either.

B.3 Conclusions

CV and time-series CV produce over-optimistic GE estimates in the presence of concept drift. Out-of-sample validation using `OOS1s` produce largely unbiased GE estimates for the immediate post-training period. However, the GE of a trained model can increase quite rapidly after the training period, which is why it is important to re-train frequently and re-estimate the GE with `OOS1s`. Using `OOS2s`, the GE in the slightly more distant future can be realistically estimated in many cases, but in some settings we observed over-optimistic GE estimates, although this over-optimism was not very strong. `OOS1s` and `OOS2s` is associated with quite a large variance, especially for small training set sizes, which is why in general it seems difficult to realistically estimate the GE for small data sets.

Note that it is not very surprising that `OOS1s` had the lowest bias, since with this procedure the training data is very similar to the entire data set on which the final model is fitted, and the test data, being at the end of the training period, is very similar to the next observations after the training period. This is also the case for other forms of concept drift beyond the incremental drift examined in the simulation, such as abrupt drift, which is why `OOS1s` should work well for these types of drift as well.

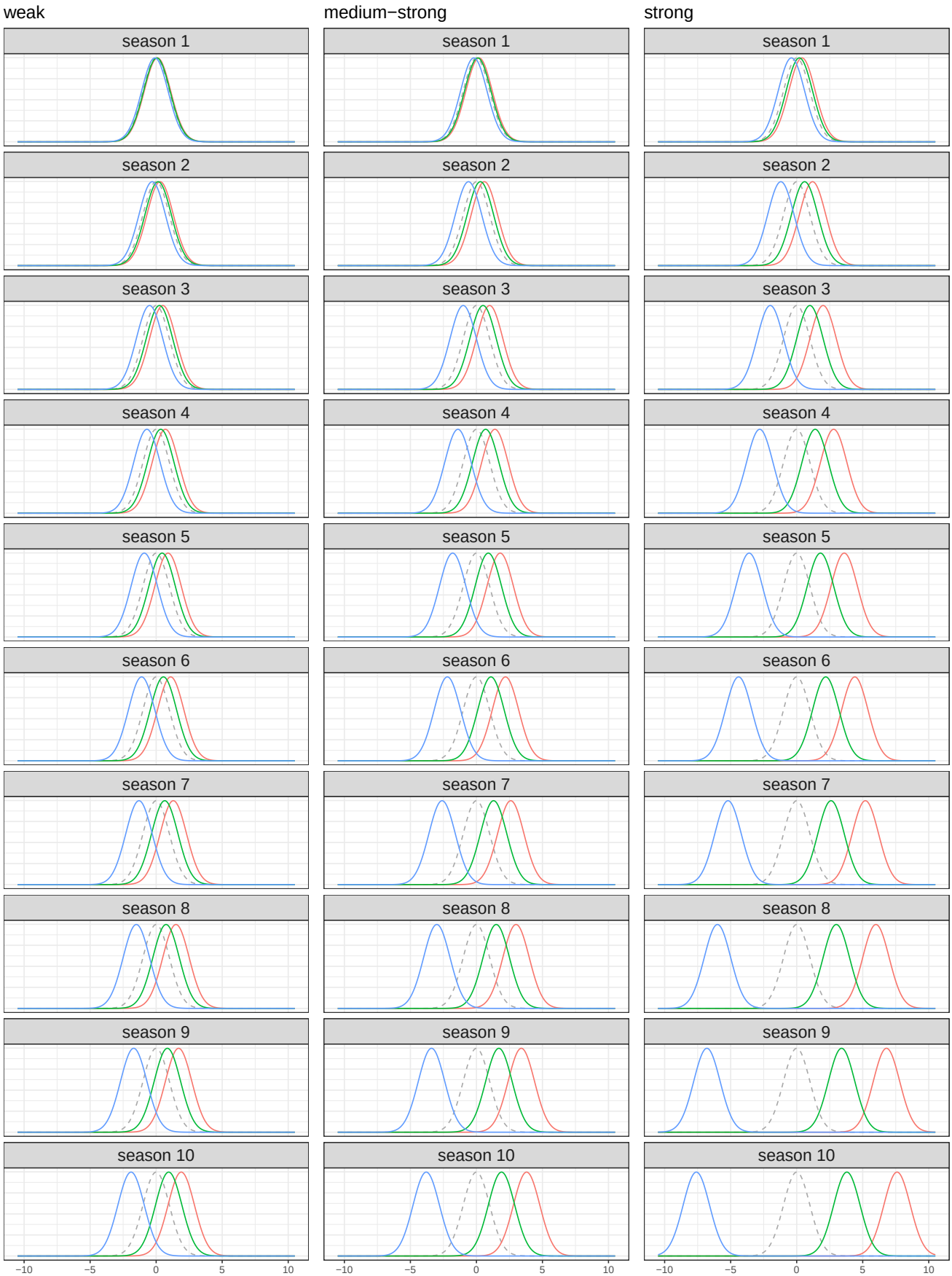

Fig. S7: Visualization of the feature drift. The grey dashed lines show the baseline distribution of the features at the beginning of season 1. The red, green, and blue solid lines show the distributions of the features $x_1^{(t)}$, $x_2^{(t)}$, and $x_3^{(t)}$, respectively, in the middles of the respective seasons.

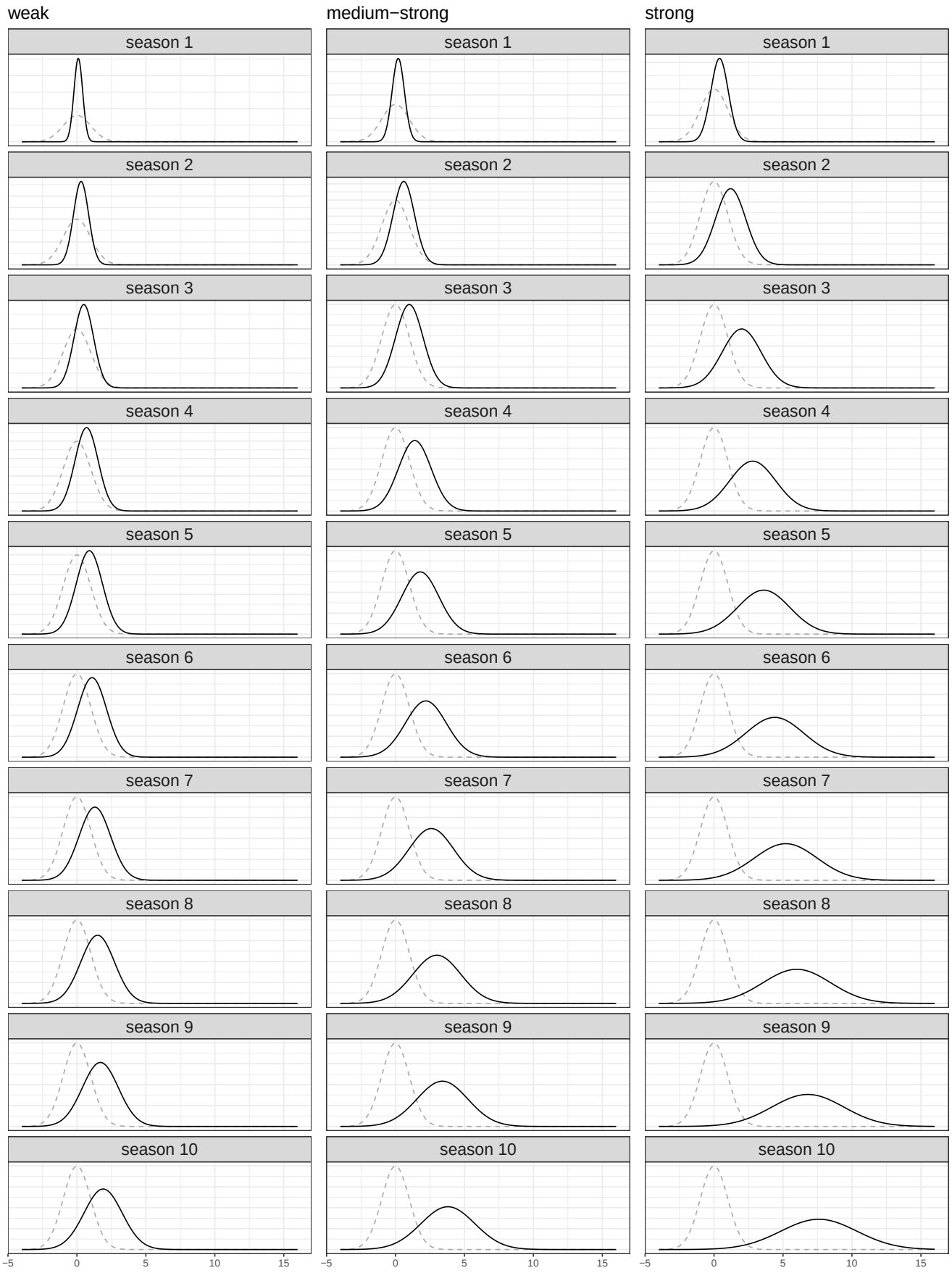

Fig. S8: Visualization of the label drift. The grey dashed lines show the baseline distribution of the label (without the influence of the features $\sum_{l=1}^{5} \beta_l x_l^{(t)}$) at the beginning of season 1. The black solid lines show the corresponding distributions in the middles of the respective seasons.

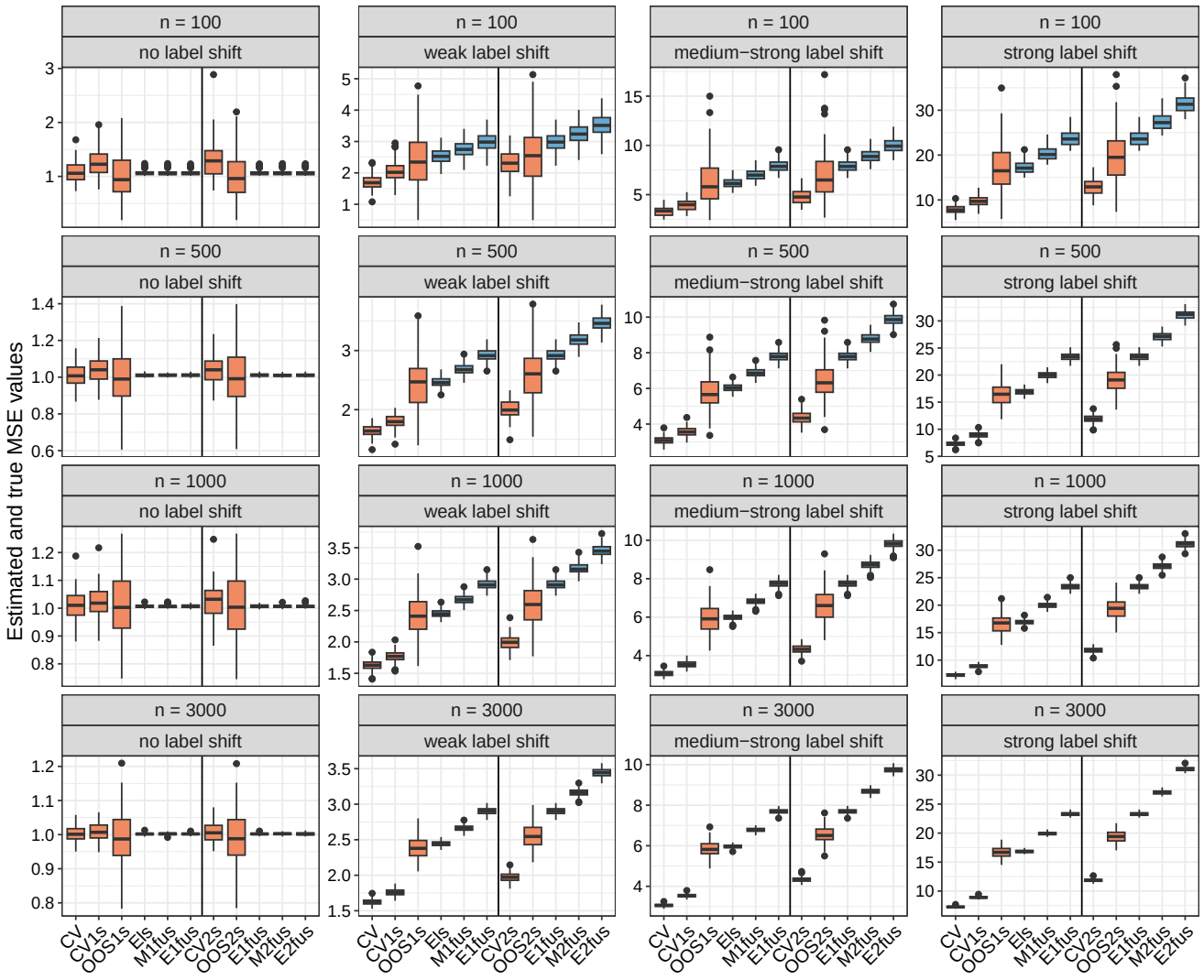

Fig. S9: Simulation on concept drift: estimated and (approximated) true MSE values of the linear models for the settings without feature shift. The orange and blue boxplots show the estimated and true MSE values, respectively. `Els`, `M1fus`, `E1fus`, `M2fus`, and `E2fus` indicate the true errors at the end of the eighth season, the middle and end of the ninth season, and the middle and end of the tenth season, respectively.

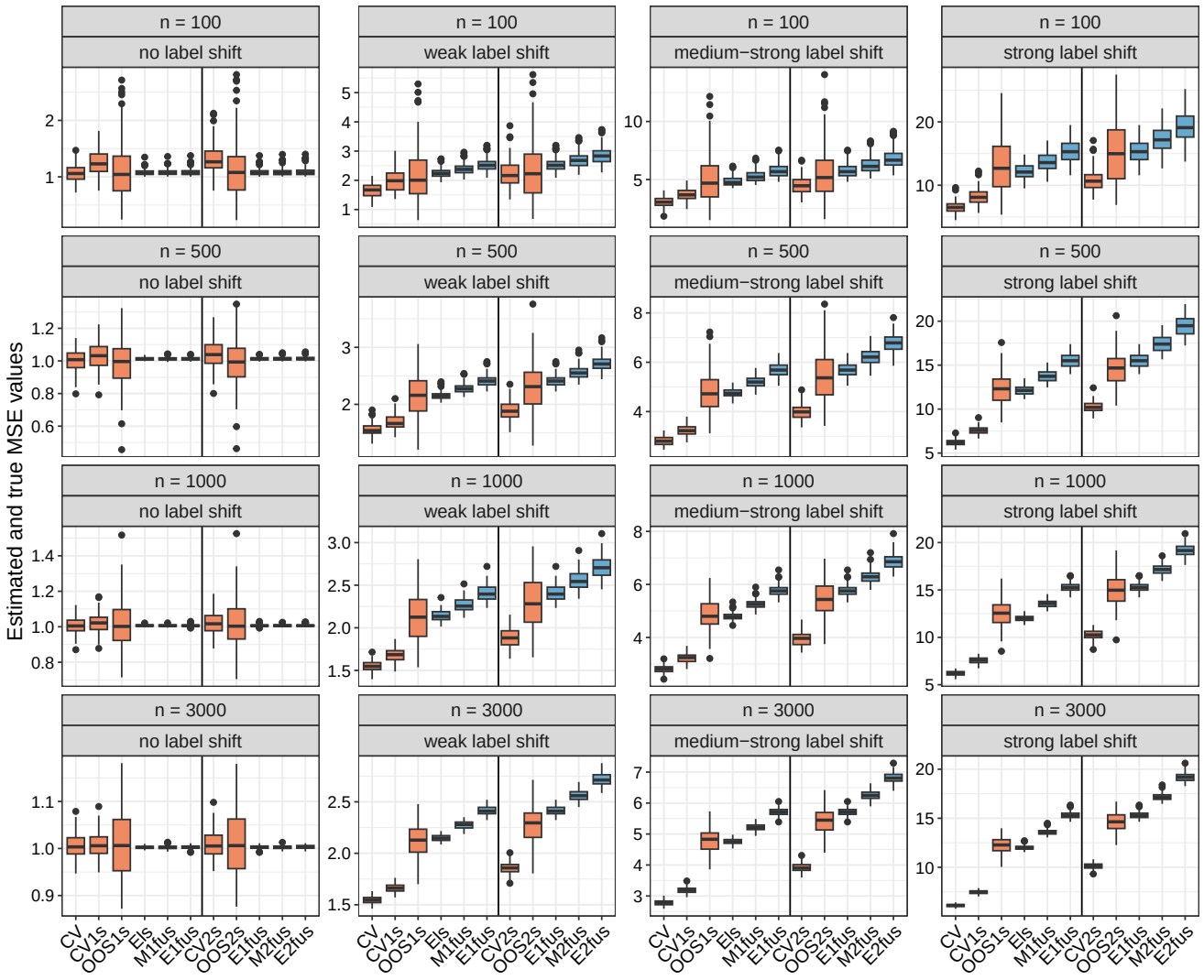

Fig. S10: Simulation on concept drift: estimated and (approximated) true MSE values of the linear models for the settings with weak feature shift. The orange and blue boxplots show the estimated and true MSE values, respectively. `Els`, `M1fus`, `E1fus`, `M2fus`, and `E2fus` indicate the true errors at the end of the eighth season, the middle and end of the ninth season, and the middle and end of the tenth season, respectively.

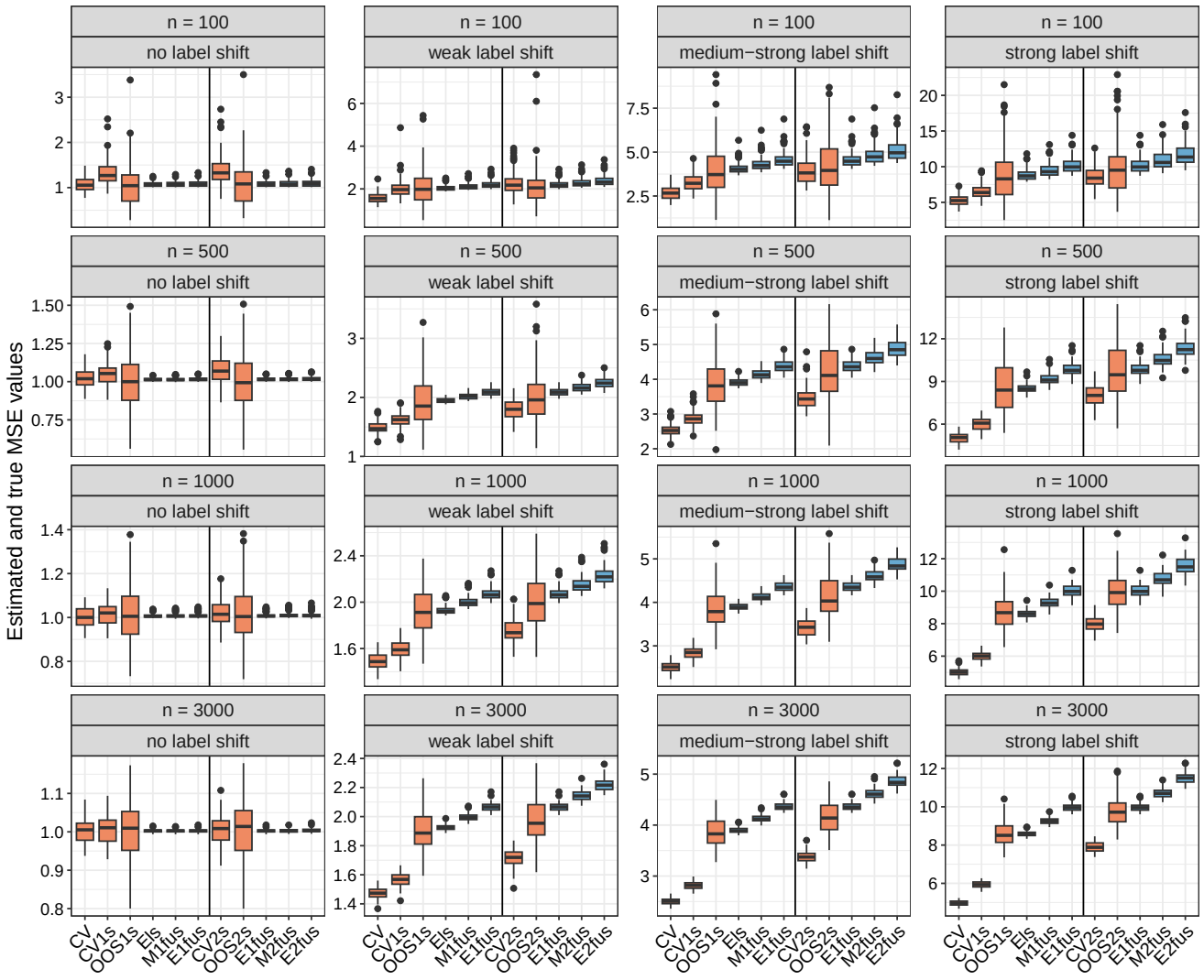

Fig. S11: Simulation on concept drift: estimated and (approximated) true MSE values of the linear models for the settings with medium-strong feature shift. The orange and blue boxplots show the estimated and true MSE values, respectively. `Els`, `M1fus`, `E1fus`, `M2fus`, and `E2fus` indicate the true errors at the end of the eighth season, the middle and end of the ninth season, and the middle and end of the tenth season, respectively.

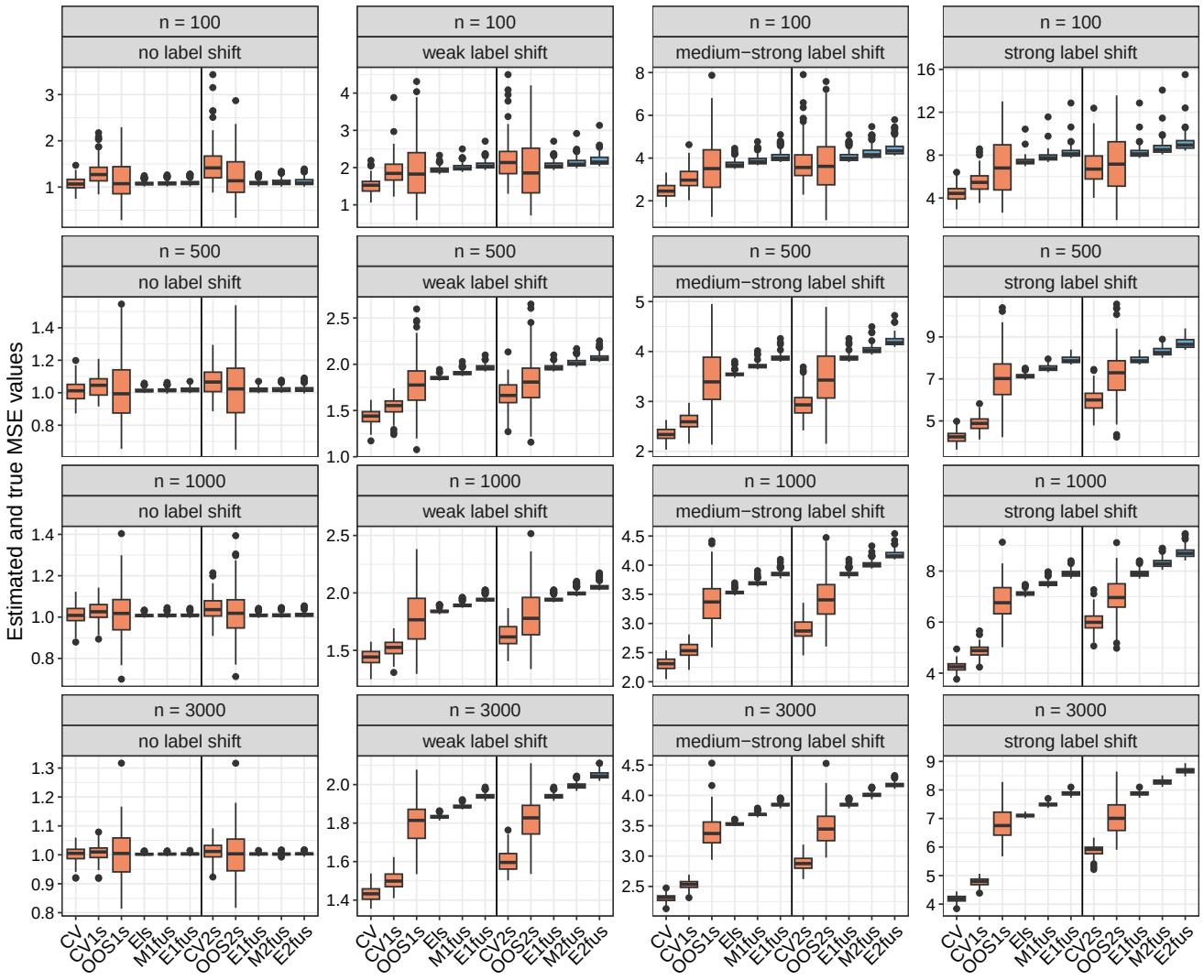

Fig. S12: Simulation on concept drift: estimated and (approximated) true MSE values of the linear models for the settings with strong feature shift. The orange and blue boxplots show the estimated and true MSE values, respectively. `Els`, `M1fus`, `E1fus`, `M2fus`, and `E2fus` indicate the true errors at the end of the eighth season, the middle and end of the ninth season, and the middle and end of the tenth season, respectively.

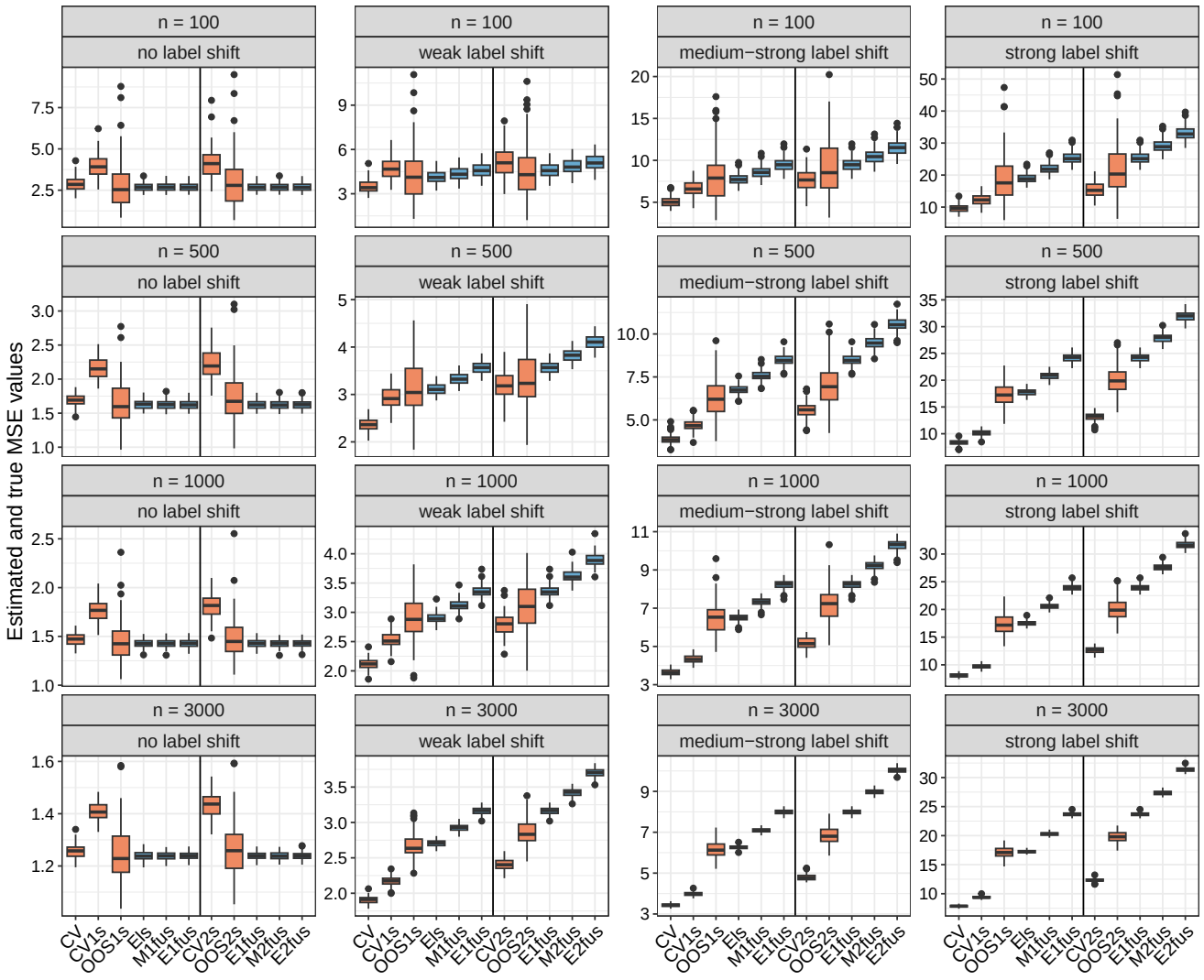

Fig. S13: Simulation on concept drift: estimated and (approximated) true MSE values of the random forests for the settings without feature shift. The orange and blue boxplots show the estimated and true MSE values, respectively. `Els`, `M1fus`, `E1fus`, `M2fus`, and `E2fus` indicate the true errors at the end of the eighth season, the middle and end of the ninth season, and the middle and end of the tenth season, respectively.

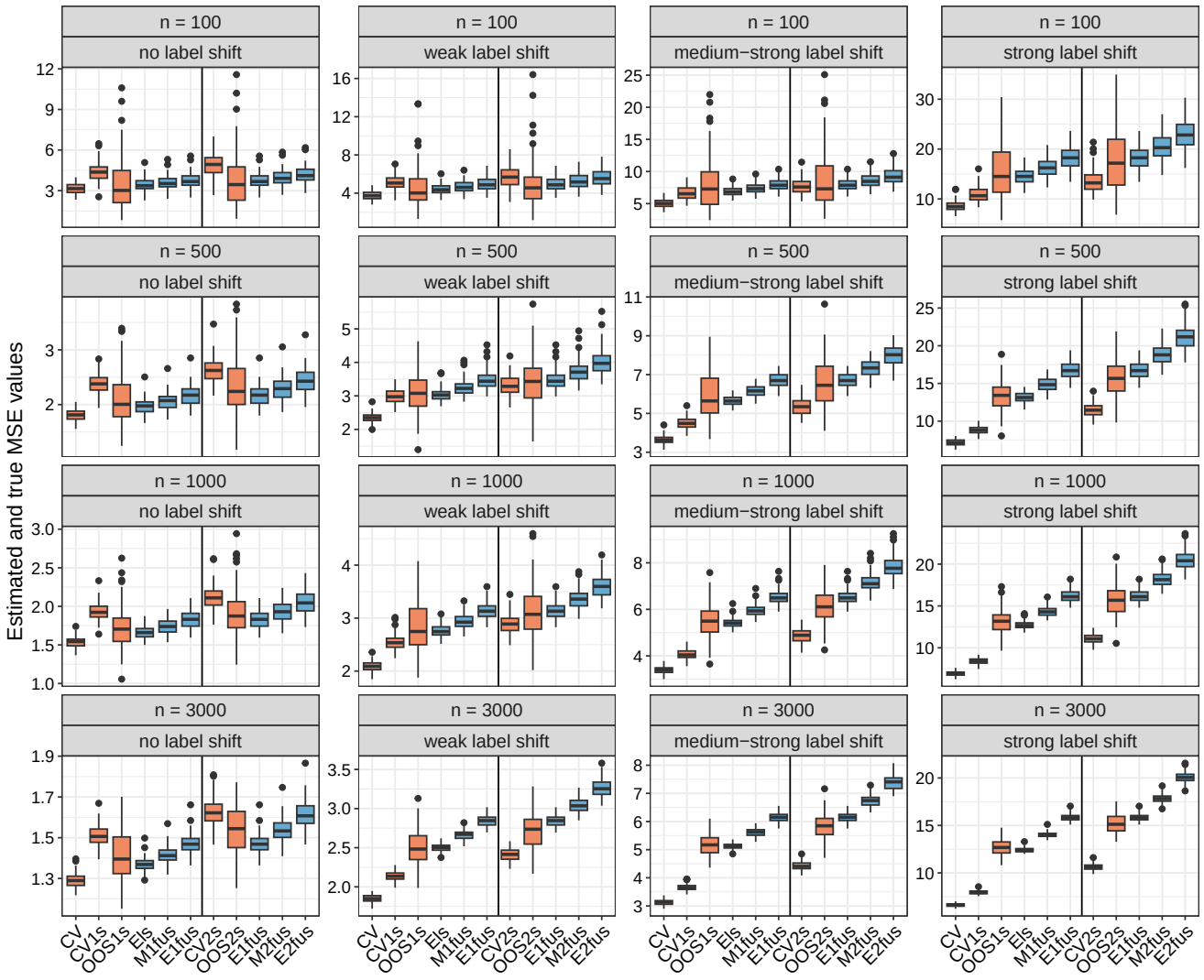

Fig. S14: Simulation on concept drift: estimated and (approximated) true MSE values of the random forests for the settings with weak feature shift. The orange and blue boxplots show the estimated and true MSE values, respectively. `Els`, `M1fus`, `E1fus`, `M2fus`, and `E2fus` indicate the true errors at the end of the eighth season, the middle and end of the ninth season, and the middle and end of the tenth season, respectively.

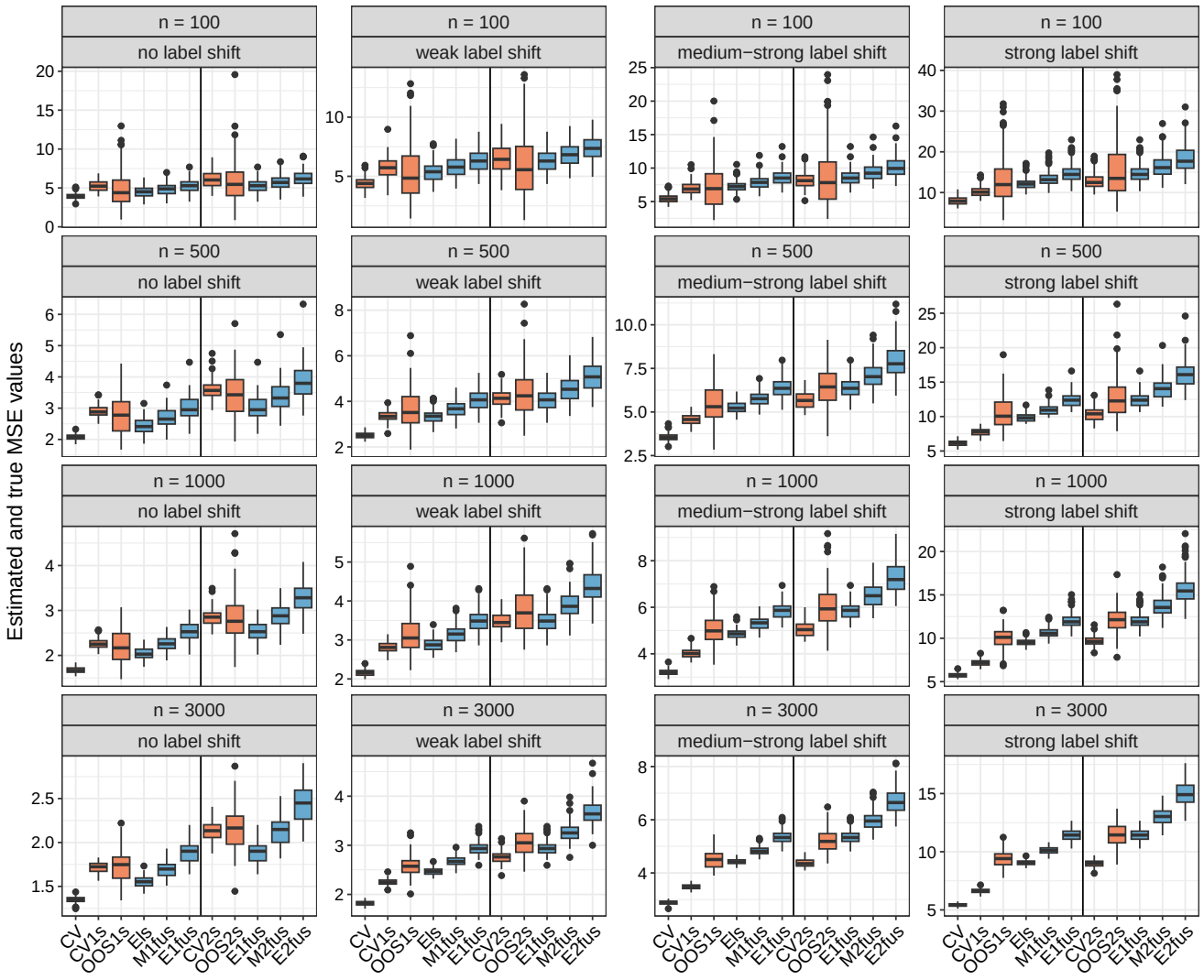

Fig. S15: Simulation on concept drift: estimated and (approximated) true MSE values of the random forests for the settings with medium-strong feature shift. The orange and blue boxplots show the estimated and true MSE values, respectively. `Els`, `M1fus`, `E1fus`, `M2fus`, and `E2fus` indicate the true errors at the end of the eighth season, the middle and end of the ninth season, and the middle and end of the tenth season, respectively.

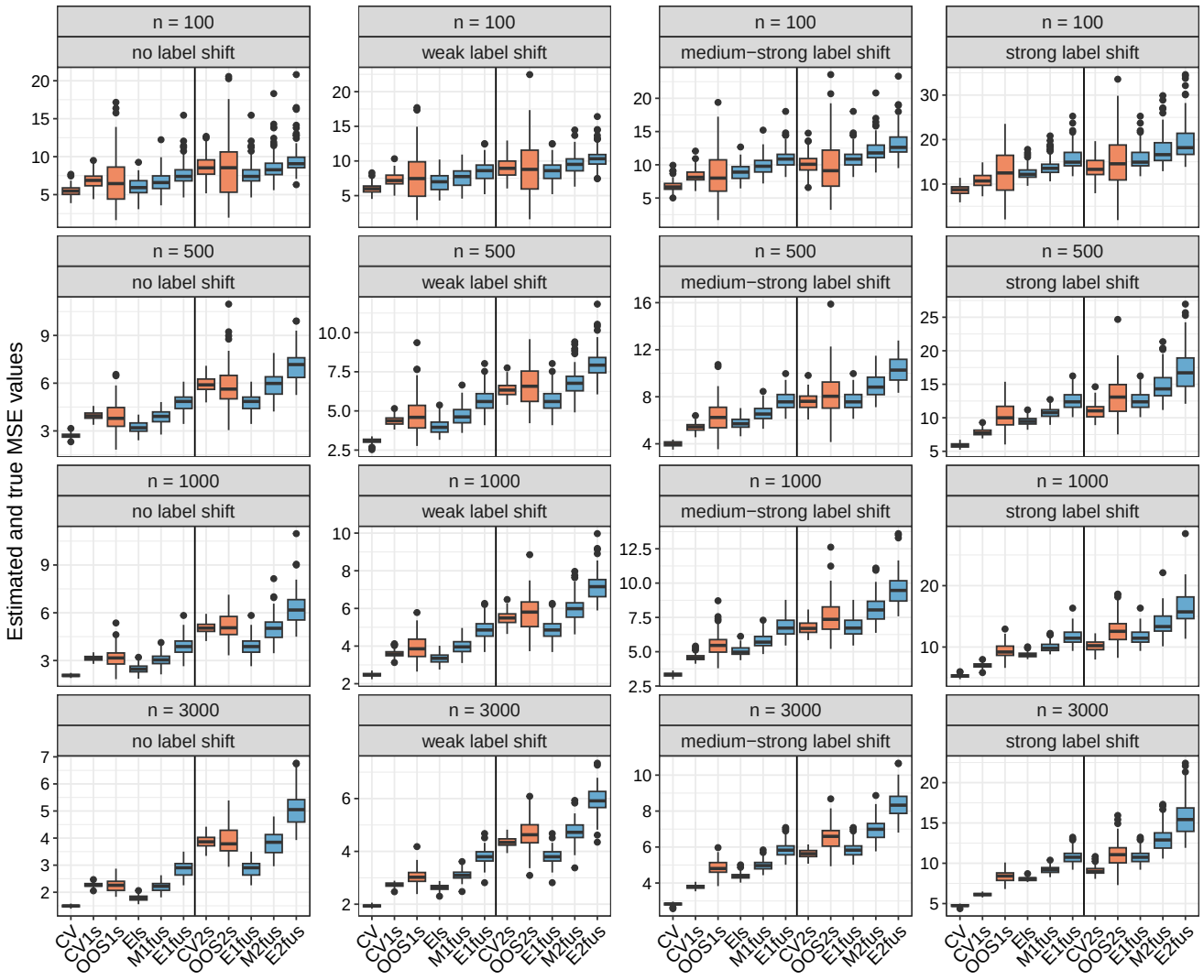

Fig. S16: Simulation on concept drift: estimated and (approximated) true MSE values of the random forests for the settings with strong feature shift. The orange and blue boxplots show the estimated and true MSE values, respectively. `Els`, `M1fus`, `E1fus`, `M2fus`, and `E2fus` indicate the true errors at the end of the eighth season, the middle and end of the ninth season, and the middle and end of the tenth season, respectively.

C Simulation study comparing stratified with non-stratified CV for hierarchical classification problems: Full design and extensive results

C.1 Simulation design

C.1.1 General proceeding

We compared standard CV with stratified CV (SCV) with respect to their performance in estimating the following nine evaluation metrics: accuracy (`acc`), micro-averaged / macro-averaged hierarchical precision (`hierpr_micro` / `hierpr_macro`), micro-averaged / macro-averaged hierarchical recall (`hierre_micro` / `hierre_macro`), micro-averaged / macro-averaged hierarchical F_1 score (`hierf_micro` / `hierf_macro`), weighted shortest path loss measure (`spath`), H-loss (`hloss`). More precisely, we performed (stratified) 5-fold CV, repeated ten times. As a classifier, we used top-down hierarchical classification (see e.g. Naik and Rangwala (2018)) with random forests as local classifiers, implemented in our R package `hierclass` available on GitHub (version 0.1.0, link: `http://github.com/RomanHornung/hierclass`). For each random forest, we constructed 500 trees.

In each simulate repetition, we simulated a training data set $\mathcal{D}_{\text{train}}$ of size n_{train} and a very large test data set $\mathcal{D}_{\text{test}}$ of size $n_{\text{test}} = 200,000$. Subsequently, first, the evaluation metric values were estimated using (stratified) CV on $\mathcal{D}_{\text{train}}$. Second, the classifier was trained on $\mathcal{D}_{\text{train}}$ and the true values of the evaluation metrics approximated using $\mathcal{D}_{\text{test}}$.

We considered four training set sizes $n_{\text{train}} \in \{200, 500, 1000, 3000\}$ and performed 100 simulation repetitions for each of these.

C.1.2 Data-generating process

Supplementary Figure S17 shows the category tree used in the simulation. Each node in this tree was simulated to have two child nodes with probability 2/3 and three child nodes with probability 1/3. We randomly generated category trees until we obtained a tree with exactly 50 leaf nodes, which is the tree shown in Supplementary Figure S17.

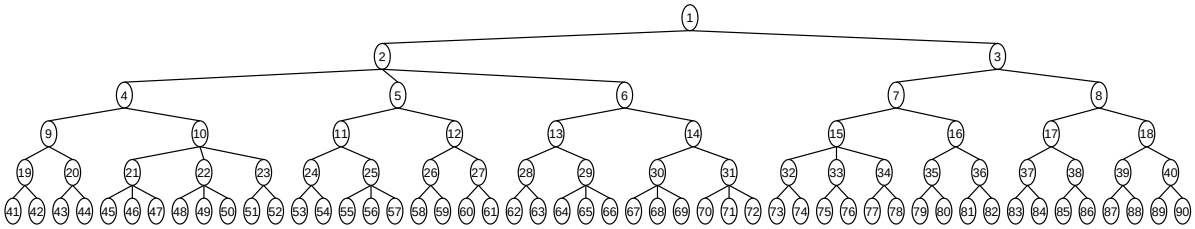

Fig. S17: Category tree used in the simulations study.

There were five features $\mathbf{x}^{(i)} = (x_1^{(i)}, \ldots, x_5^{(i)})^T$, which were all drawn from $\mathcal{N}(0, 1)$. To assign the hierarchical classes to the observations, the latter were dropped down the category tree, where in each node a multinomial logistic regression model was used to decide for the next child node in the tree hierarchy:

$$\pi_{m1i} = \frac{1}{1 + \sum_{s=2}^{c_m} \exp(\beta_{sm0} + \mathbf{x}^{(i)T}\boldsymbol{\beta}_{sm})}, \quad \pi_{mri} = \frac{\beta_{rm0} + \exp(\mathbf{x}^{(i)T}\boldsymbol{\beta}_{rm})}{1 + \sum_{s=2}^{c_m} \exp(\beta_{sm0} + \mathbf{x}^{(i)T}\boldsymbol{\beta}_{sm})}, \quad r \in \{2, 3\}.$$

Here, π_{mri}, $r \in \{1, 2, 3\}$ denotes the probability that observation i is assigned to the r-th child node and m indexes the current node. Note that this description applies to nodes with three child nodes, but the case of two child nodes was very similar with the only difference being that $r \in \{1, 2\}$. The parameters β_{rm0} and $\boldsymbol{\beta}_{rm} = (\beta_{rm1}, \ldots, \beta_{rm5})^T$,

$r \in \{2, 3\}$, were sampled as follows:

$$\beta_{rm0} \sim \mathcal{N}(0, 1), \quad \beta_{rmj} \sim \mathcal{N}(0, \sigma_l^2), \quad r \in \{2, 3\}, \quad j \in \{1, \ldots, 5\}.$$

Here, the variance σ_l^2 of the feature coefficients is indexed by the layer l of the node. These values were set as follows: $\sigma_1^2 := 2.5$, $\sigma_2^2 := 2$, $\sigma_3^2 := 0.9$, $\sigma_4^2 := 0.7$, and $\sigma_5^2 := 0.5$. Choosing smaller values of σ_l^2 for lower layers in the category tree had the effect or reducing the influence of the input features in lower layers. This in turn made the data in the child nodes more similar in lower layers, which is a common feature of hierarchical classification problems.

In each simulation repetition, we first generated the parameter values β_{rm0} and $\boldsymbol{\beta}_{rm}$ for all nodes in the category tree. Subsequently, using these parameter values we simulated the training data set $\mathcal{D}_{\text{train}}$ and the test data set $\mathcal{D}_{\text{test}}$.

C.2 Results

Supplementary Figure S18 shows the estimated and true evaluation metric values. As expected, for most evaluation metrics the performance got better with larger training set sizes. Exceptions are the macro-averaged metrics, where only the true performance got better with larger training set sizes and even here only for `hierre_macro` and `hierf_macro`, but not `hierpr_macro`.

One striking observation to be made is that, for $n_{\text{train}} = 200$, both CV versions severely overestimated the true `hierre_macro` values. Important reasons for this over-optimism are the facts that smaller classes tended to receive much worse class-specific hierarchical recall values and that (stratified) CV implicitly assigned less weight to smaller classes in the estimation of the `hierre_macro`. The reason why small classes tended to receive much worse class-specific hierarchical recall values was that many prediction methods, random forests in particular, have a strong tendency towards predicting larger classes. And the reason why CV implicitly assigned less weight to smaller classes in the estimation of `hierre_macro` was that small classes frequently did not occur in the test folds in CV, in which cases these classes were not included in the estimation of the `hierre_macro` values on these test folds. This mechanism took effect for SCV as well even though here the folds were balanced with respect to the class distributions. This is because, for $n_{\text{train}} = 200$, small classes often only featured very few observations (< 5), which had the effect that only a subset of the folds featured these classes. Another reason for the large discrepancy between estimated and true `hierre_macro` values was that, for $n_{\text{train}} = 200$, many small classes did not even occur in the training sets and these classes naturally tend to receive very bad class-specific hierarchical recall values in the approximation of the true `hierre_macro` values.

We observed much less over-optimism for estimating the true `hierpr_macro` values. There are several reasons for this. First, given the strong tendency of random forests to predict larger classes, many of the smallest classes were not predicted at all, even on the huge test sets ($n_{\text{test}} = 200,000$) used for approximating the true `hierpr_macro` values (results not shown), which is why the true `hierpr_macro` values were less influenced negatively by the smallest classes. Second, given the definition of `hierpr_macro`, in contrast to in the case of `hierre_macro`, no classes that did not occur in the training sets were considered in the class-specific hierarchical precision values in the approximation of the true values for `hierpr_macro`. This also had the effect that small classes that did not occur in the training sets did not negatively influence the approximated true `hierpr_macro` values. Lastly, the class-specific hierarchical precision values tended to be less worse for small classes compared to in the cases of the class-specific hierarchical recall values. This is because, in cases in which prediction rules predicted small classes, the chance that the predicted class was actually the true class was relatively high. The latter was a result of the fact that prediction rules predict small classes rarely, which is why in cases in which they do predict small classes, there tends to be quite much evidence for the predicted class. The over-optimism observed for `hierf_macro` can be explained by the fact that the latter is the harmonic mean of `hierre_macro` and `hierpr_macro`.

Supplementary Figure S19 shows the (standardized) differences between the evaluation metrics estimates and their true values. As already seen in Supplementary Figure S18 these differences tended to be relatively small with the exception of in the cases of the macro-averaged metrics and the smallest training set size $n_{\text{train}} = 200$. For

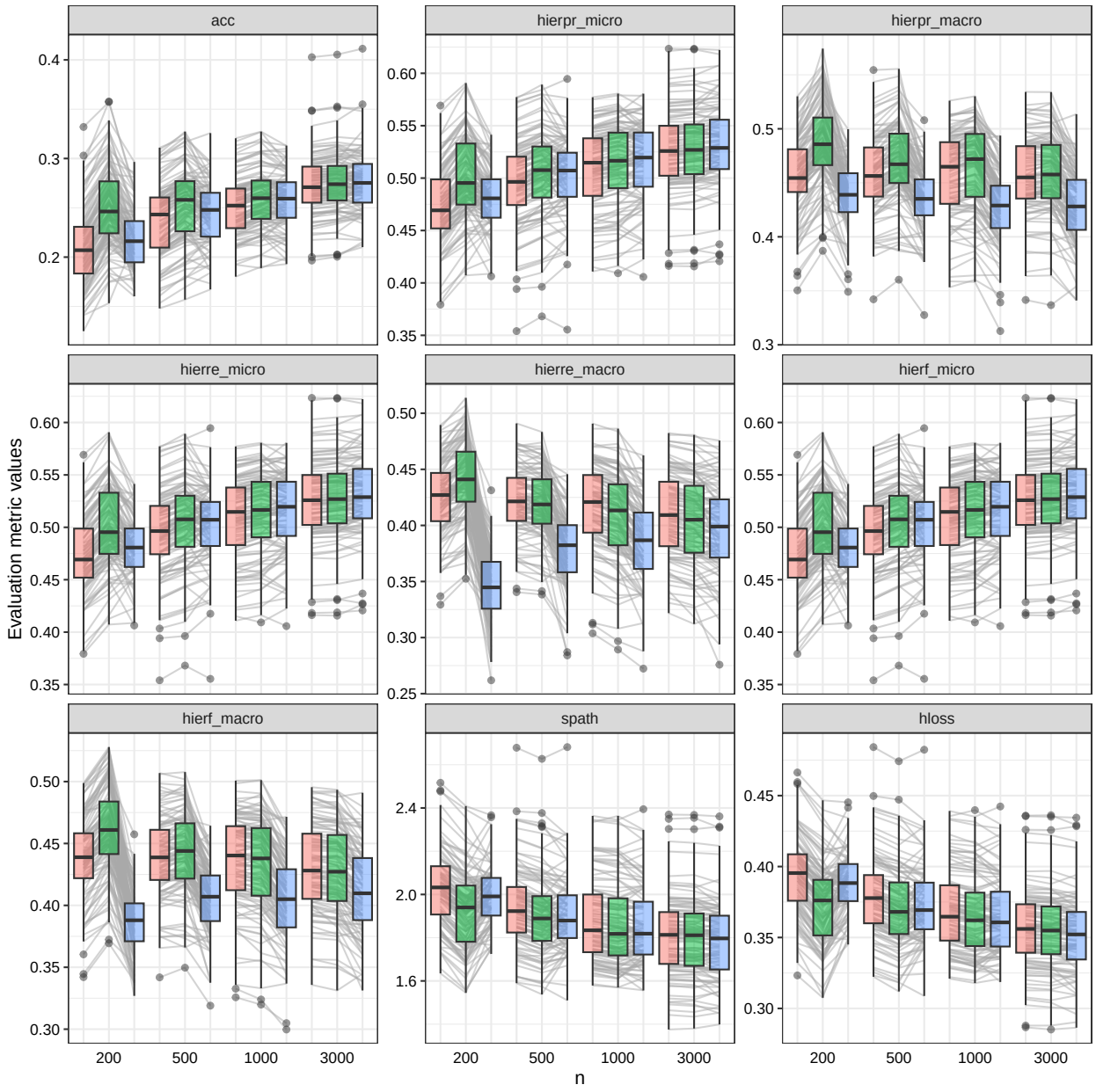

Fig. S18: Simulation on hierarchical classification: estimated and (approximated) true evaluation metric values. The red and green boxplots show the metric values obtained using CV and SCV, respectively, and the blue boxplots show the true metric values approximated using the very large test data sets. The grey lines connect the results obtained using the same simulation repetitions.

the latter, SCV tended to overestimate the true performance, while CV tended to underestimate it. There were no notable differences in terms of the variance of the estimates between the two CV variants. One reason for the underestimation of the performance observable for CV was the well-known inherent bias of CV. The latter is due to the training set sizes used during CV being smaller than the whole data set, which is used for training the final classifier. Note, however, that SCV was subject to this bias as well. Another likely reason for the underestimation, that is, however, specific to CV is that, for $n_{\text{train}} = 200$, the differences in class distributions between training and test folds were frequently quite strong. If a class was over-proportionally frequent in the test fold, it was under-proportionally frequent in the training fold and vice versa. As stated above, prediction methods, random forests in particular, favor larger classes, which is why the prediction performance on larger classes dominated the prediction performance estimates. If many observations of a large class were in the test fold, there were few observations in the training fold. In this case, the classifier learned on the training fold was likely not able to

predict the class well. If so, this contributed to a bad estimated predictive performance because many observations from this class were in the test fold. In contrast, if few observations from a large class were in the test fold and many in the training fold, the learned classifier was likely able to predict the observations from this class well in the test fold. However, if this was the case, it did not have a strong positive influence on the estimated predictive performance because there were only few observations from that class in the test fold in that situation.

For SCV, the folds are balanced with respect to the class distribution. Therefore, the classifiers learned within SCV can be expected to feature similar performance. One reason for the observation that the estimated evaluation metrics were more optimistic than the approximated true evaluation metrics was that for $n_{\text{train}} = 200$ many classes were contained only in the test data sets used for approximating the true values of the evaluation metrics. This was expected and not an issue with the SCV procedure because a classifier generally performs bad on unseen classes. However, even when including in the test data only observations from classes that were contained in the training data, SCV still overestimated the true performance, albeit less strongly and not for all evaluation metrics (results not shown). This was likely related to the fact that, for $n_{\text{train}} = 200$, the class distribution in the training data could differ from the class distribution in the test data through random fluctuations. With SCV, however, the class distributions in the test folds were made artificially equal to those in the training folds. This likely contributed to the observation that SCV results in over-optimistic performance estimates for $n_{\text{train}} = 200$.

For larger training set sizes, the bias associated with SCV tended to be lower than that associated with CV. For the macro-averaged metrics `hierpr_macro` and `hierf_macro`, both CV variants were associated with some level of over-optimism even for the largest training set size $n_{\text{train}} = 3000$. However, for most evaluation metrics SCV was associated with no notable bias for larger training set sizes, whereas the bias of CV prevailed in this range. For the largest training set size $n_{\text{train}} = 3000$, SCV was also associated with slight optimistic bias for most evaluation metrics, which might have been due to the inherent bias of CV. However, for $n_{\text{train}} = 3000$ the bias was negligible for both CV variants.

In summary, for very small training set sizes, SCV seems to be associated with a slight over-optimistic bias and CV with a slight pessimistic bias. For larger training set sizes SCV, in contrast to CV, seems to be associated with no notable bias. However, the bias of CV tends to become smaller for larger training set sizes. In terms of the variance of the estimation, there were no notable differences between the two CV variants.

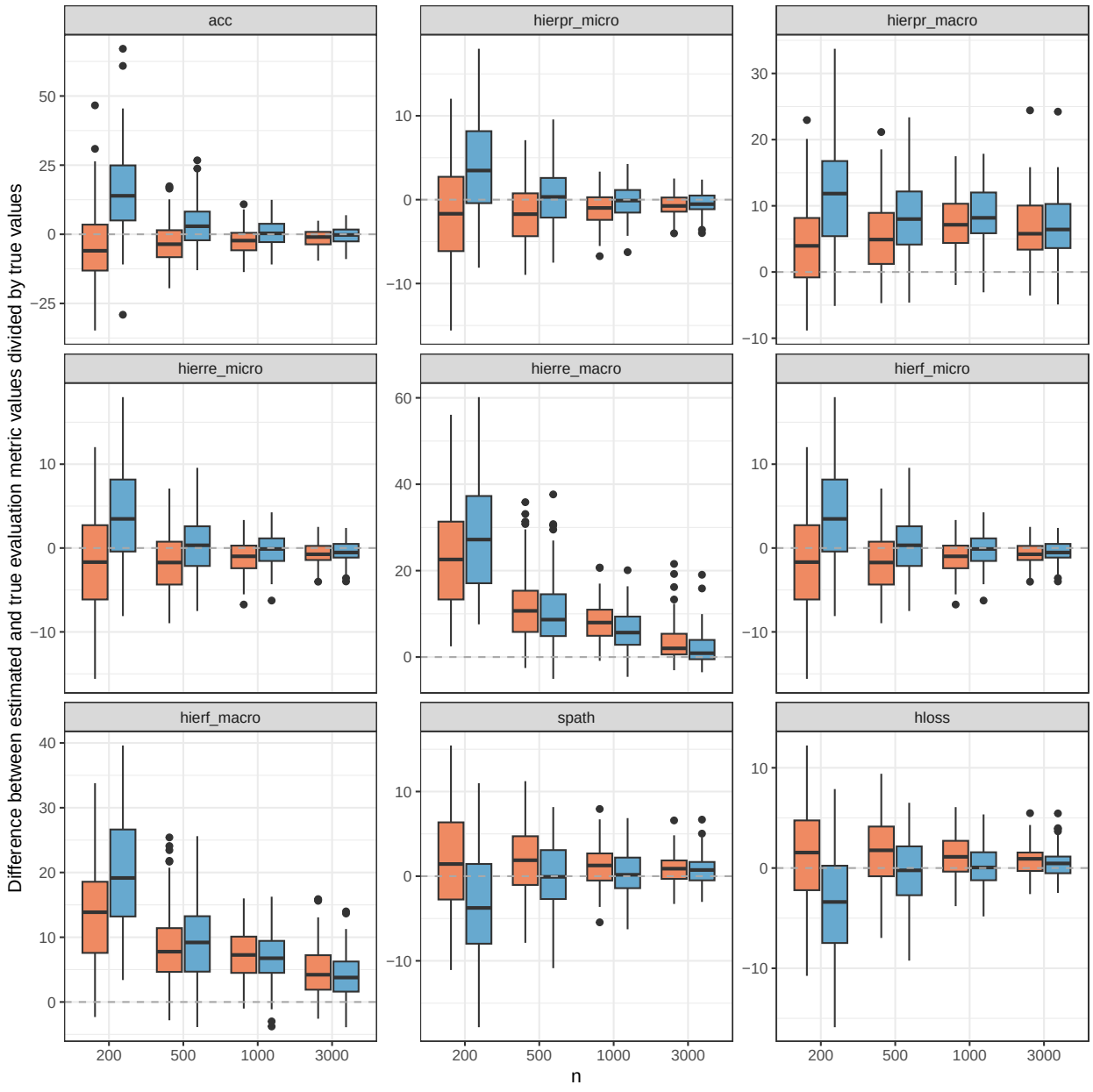

Fig. S19: Simulation on hierarchical classification: differences between estimated and true evaluation metric values divided by true values. The orange and the blue boxplots indicate the results obtained when using CV and SCV, respectively.

References

C. Bergmeir, R. J. Hyndman, and B. Koo. A note on the validity of cross-validation for evaluating autoregressive time series prediction. *Comput. Stat. Data An.*, 120:70–83, 2018. doi: 10.1016/j.csda.2017.11.003.

A. Naik and H. Rangwala. *Large Scale Hierarchical Classification: State of the Art.* Springer, Berlin, 2018. doi: 10.1007/978-3-030-01620-3.