Supplementary Material 1 for the article:

Ordinal Forests

Roman Hornung*[1]

[1] Institute for Medical Information Processing, Biometry and Epidemiology, University of Munich, Marchioninistr. 15, D-81377 Munich, Germany

*Corresponding author: e-mail: hornung@ibe.med.uni-muenchen.de, Phone: +4989-440074497, Fax: +4989-70957491

A Algorithm used in R package ordinalForest for generating the class width sets

As mentioned in the description of the OF algorithm in section 2.1 of the main paper it is important that the collection of sets $\{d_{b,1}, \ldots, d_{b,J+1}\}$ ($b \in \{1, \ldots, B_{\text{sets}}\}$) is heterogeneous enough across the iterations $1, \ldots, B_{\text{sets}}$ to ensure that the best of the considered sets $\{d_{b,1}, \ldots, d_{b,J+1}\}$ feature an OOB prediction performance close to that of the best possible set.

The following algorithm is used in the R package `ordinalForest` for generating the collection of sets $\{d_{b,1}, \ldots, d_{b,J+1}\}$ ($b \in \{1, \ldots, B_{\text{sets}}\}$):

1. In this step the rankings of the class widths for each of the B_{sets} sets are generated.

   If $J! < B_{\text{sets}}$ this is performed as follows:

   1.1. Generate all $J!$ possible permutations of $1, \ldots, J$ and permute this set of permutations randomly. The result of the latter step is the rankings $\{r_{b,1}, \ldots, r_{b,J}\}$, $b = 1, \ldots, J!$, for the first $J!$ sets.

   1.2. Copy each of the rankings $\{r_{b,1}, \ldots, r_{b,J}\}$, $b = 1, \ldots, J!$, $\lfloor B_{\text{sets}}/J! \rfloor - 1$ times to produce the rankings $\{r_{b,1}, \ldots, r_{b,J}\}$, $b = J!+1, \ldots, J!\lfloor B_{\text{sets}}/J! \rfloor$, for the next $J!(\lfloor B_{\text{sets}}/J! \rfloor - 1)$ sets.

   1.3. The last rankings $\{r_{b,1}, \ldots, r_{b,J}\}$, $b = J!\lfloor B_{\text{sets}}/J! \rfloor + 1, \ldots, B_{\text{sets}}$, are produced through random sampling from the first $J!$ rankings $\{r_{b,1}, \ldots, r_{b,J}\}$, $b = 1, \ldots, J!$.

   If $J! \geq B_{\text{sets}}$ the generation of the rankings of the class widths for each set is performed as follows:

   1.1. Permute $1, \ldots, J$ randomly once to produce the ranking $\{r_{1,1}, \ldots, r_{1,J}\}$ for the first set.

   1.2. For $b = 2, \ldots, B_{\text{sets}}$: Draw N_{perm} (e.g., $N_{\text{perm}} = 500$) random permutations of $1, \ldots, J$ denoted as $\{r^*_{l,1}, \ldots, r^*_{l,J}\}$, $l = 1, \ldots, N_{\text{perm}}$. Determine that permutation from $\{r^*_{l,1}, \ldots, r^*_{l,J}\}$, $l = 1, \ldots, N_{\text{perm}}$ that features the greatest quadratic distance to $\{r_{b-1,1}, \ldots, r_{b-1,J}\}$, that is $\arg\max_{\{r^*_{l,1}, \ldots, r^*_{l,J}\}} \sum_{j=1}^{J} (r^*_{l,j} - r_{b-1,j})^2$ and use this permutation as the ranking for the bth set.

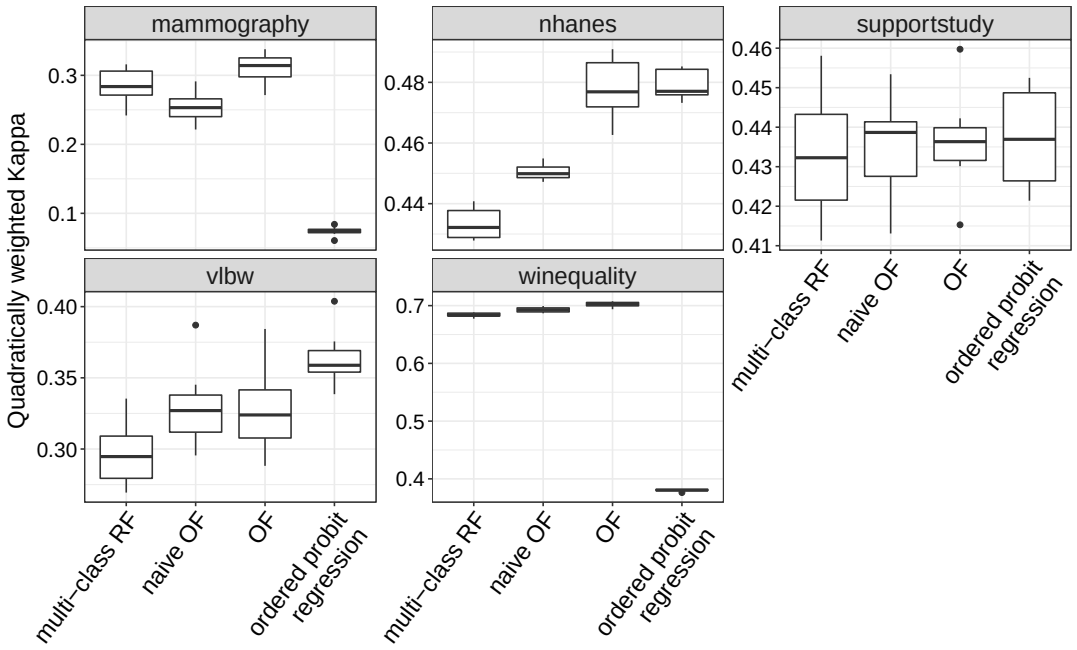
2. In this step the sets $\{d_{b,1}, \ldots, d_{b,J+1}\}$ for all iterations $b = 1, \ldots, B_{\text{sets}}$ are generated.

   For $b = 1, \ldots, B_{\text{sets}}$:

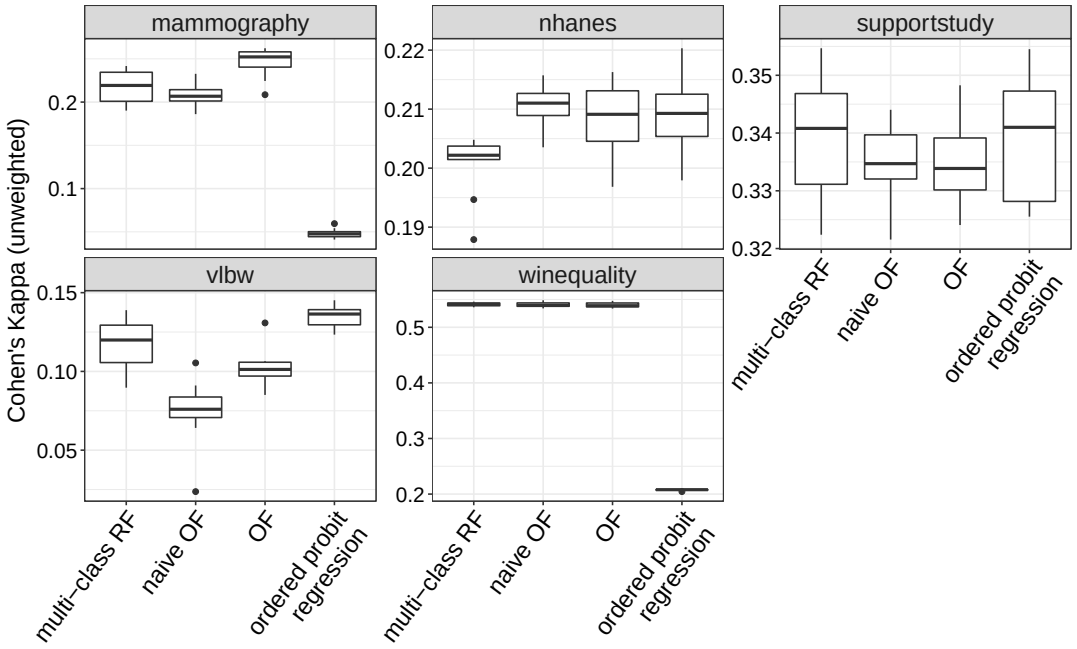
   2.1. Draw $J - 1$ instances of a $U(0, 1)$ distributed random variable and sort the resulting values. The sorted values are designated as $d^*_{b,2}, \ldots, d^*_{b,J}$. Moreover, set $d^*_{b,1} := 0$ and $d^*_{b,J+1} := 1$.

   2.2. Re-order the intervals of the $[0, 1]$ partition $\{d^*_{b,1}, \ldots, d^*_{b,J+1}\}$ in such a way that the jth interval, $j = 1, \ldots, J$, is that interval out of $]d^*_{b,1}, d^*_{b,2}], \ldots,]d^*_{b,J}, d^*_{b,J+1}]$ that features the $r_{b,j}$th smallest width and use this re-ordered partition as the bth set $\{d_{b,1}, \ldots, d_{b,J+1}\}$.

B Supplementary Figures: real data analysis - (weighted) Kappa values

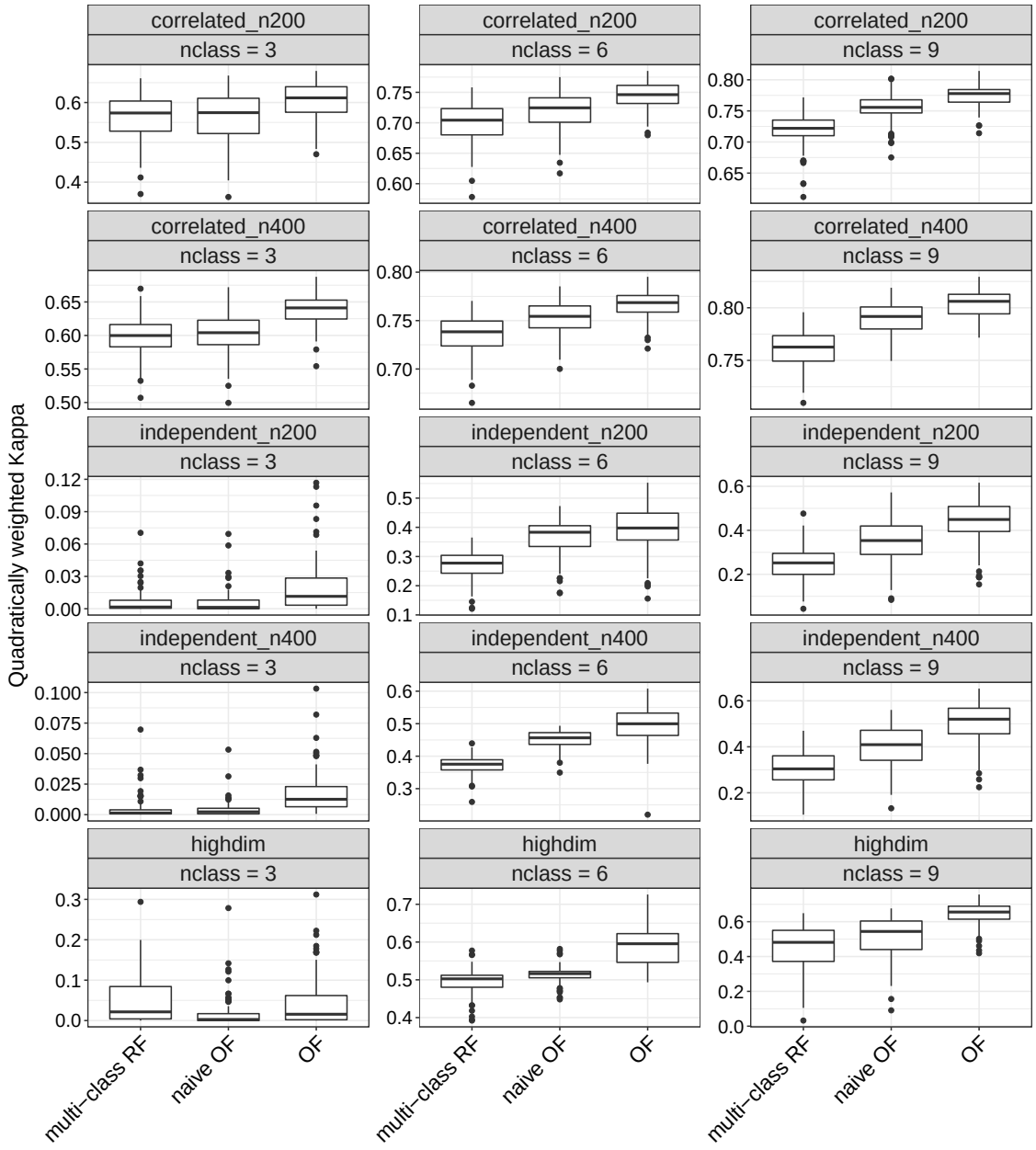


Supplementary Figure 1: Values of quadratically weighted Kappa for each of the five datasets and each of the four methods considered. Each boxplot shows the values obtained for the individual repetitions of the 10-fold stratified cross-validation.

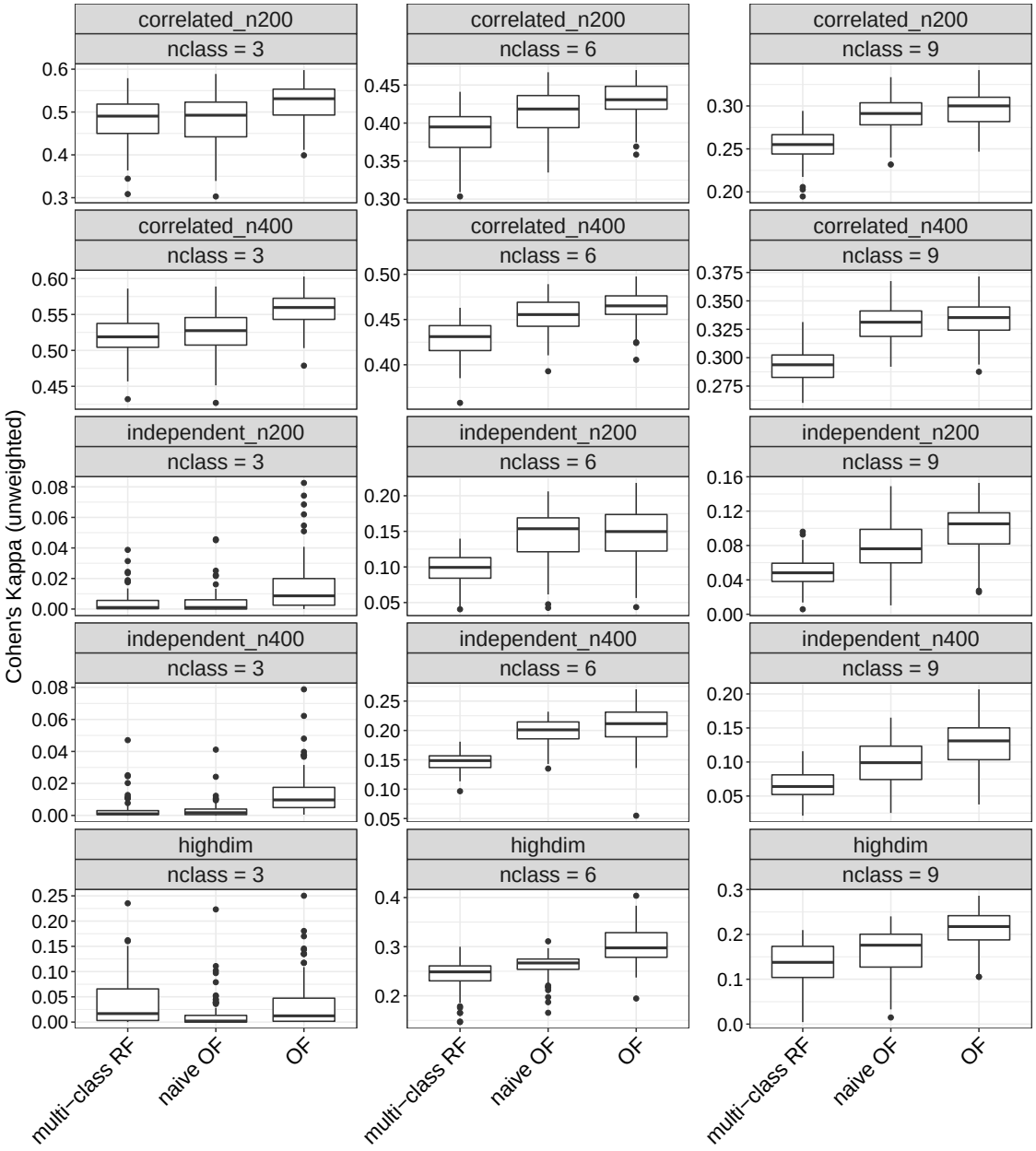


Supplementary Figure 2: Values of Cohen's Kappa for each of the five datasets and each of the four methods considered. Each boxplot shows the values obtained for the individual repetitions of the 10-fold stratified cross-validation.

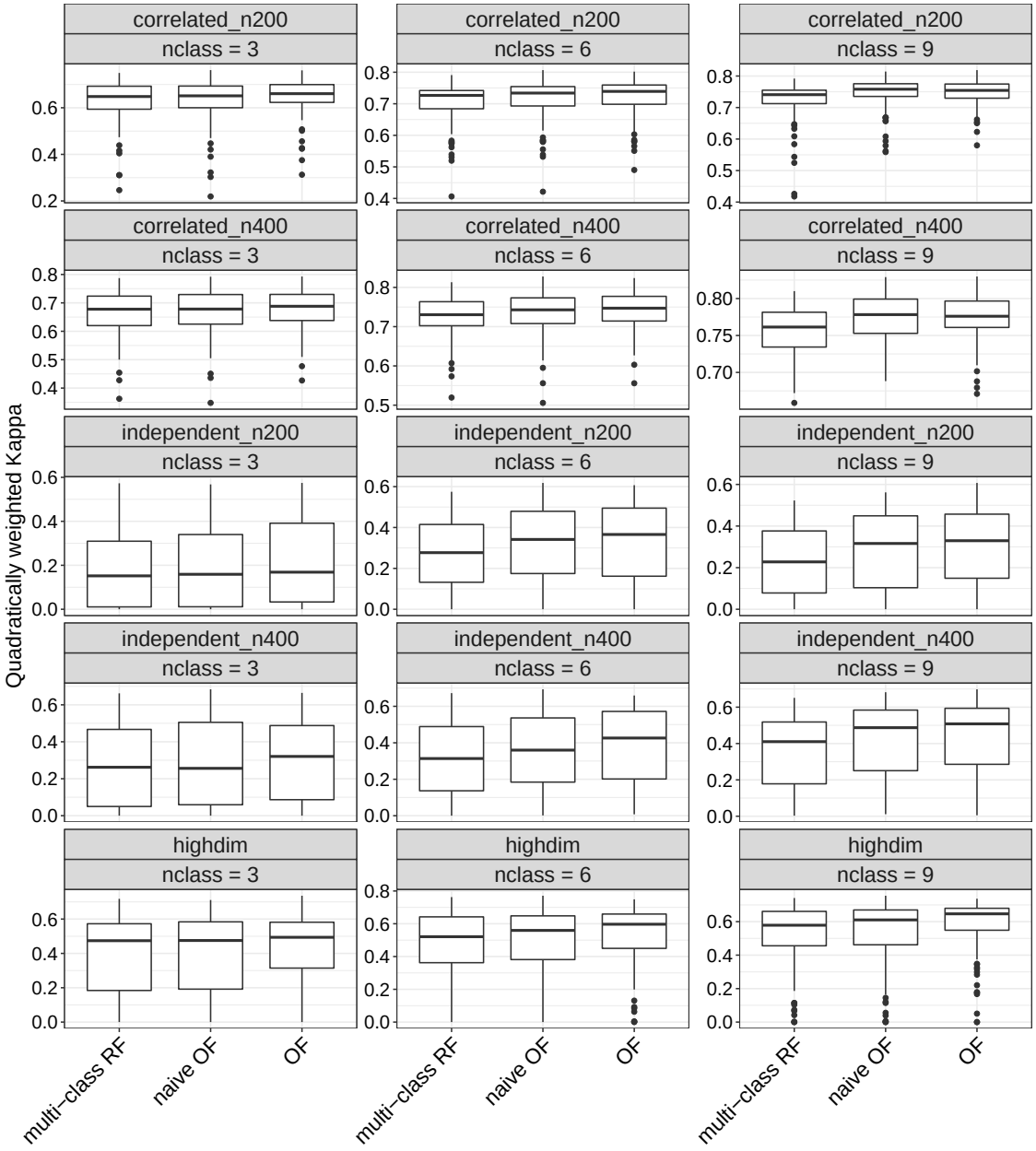
C Supplementary Figures: simulation - (weighted) Kappa values



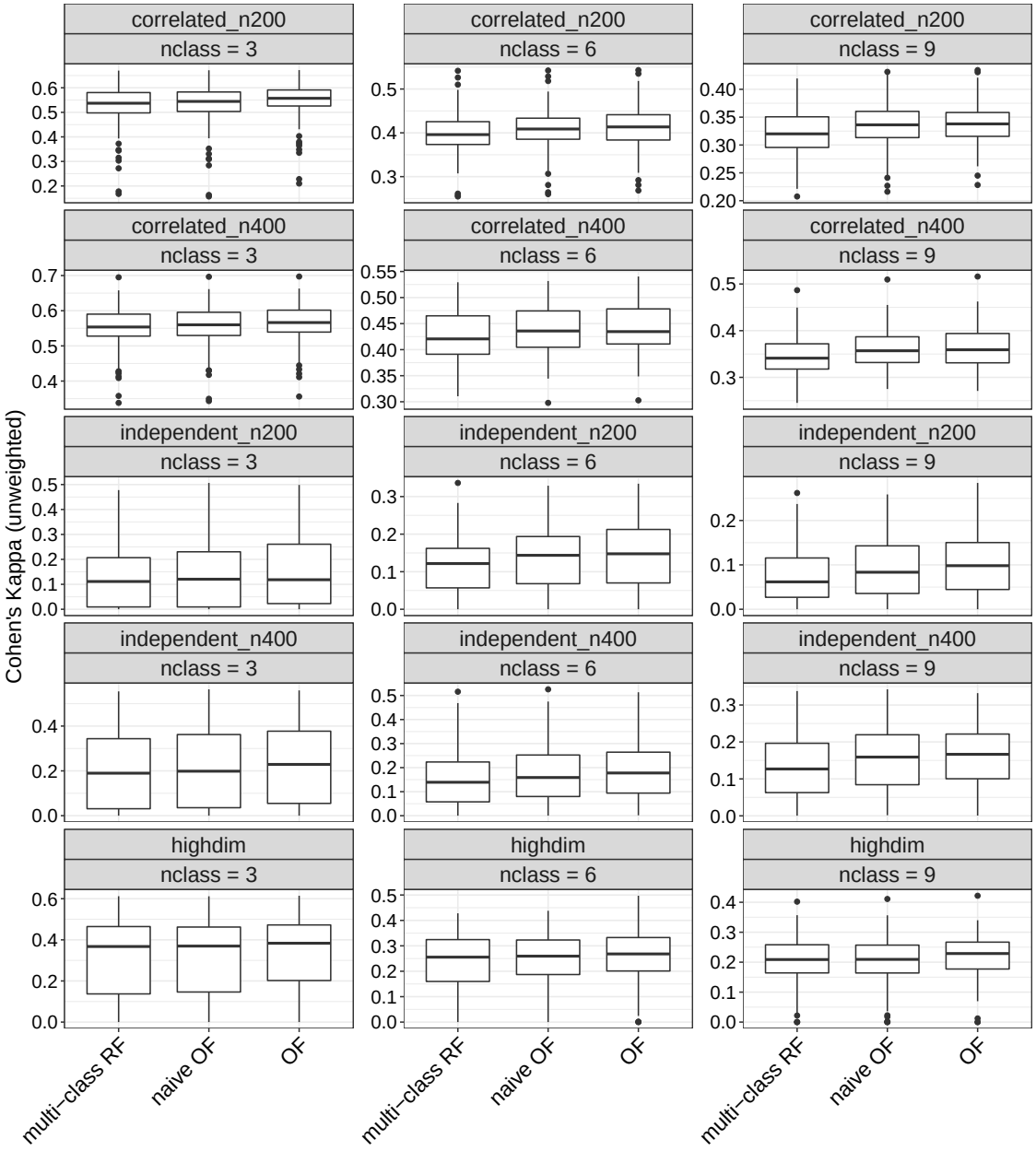
Supplementary Figure 3: Values of quadratically weighted Kappa for each simulation setting with equal class widths and each of the three methods considered. Each boxplot shows the values obtained on the corresponding test dataset for each of the 100 simulation iterations.



Supplementary Figure 4: Values of Cohen's Kappa for each simulation setting with equal class widths and each of the three methods considered. Each boxplot shows the values obtained on the corresponding test dataset for each of the 100 simulation iterations.

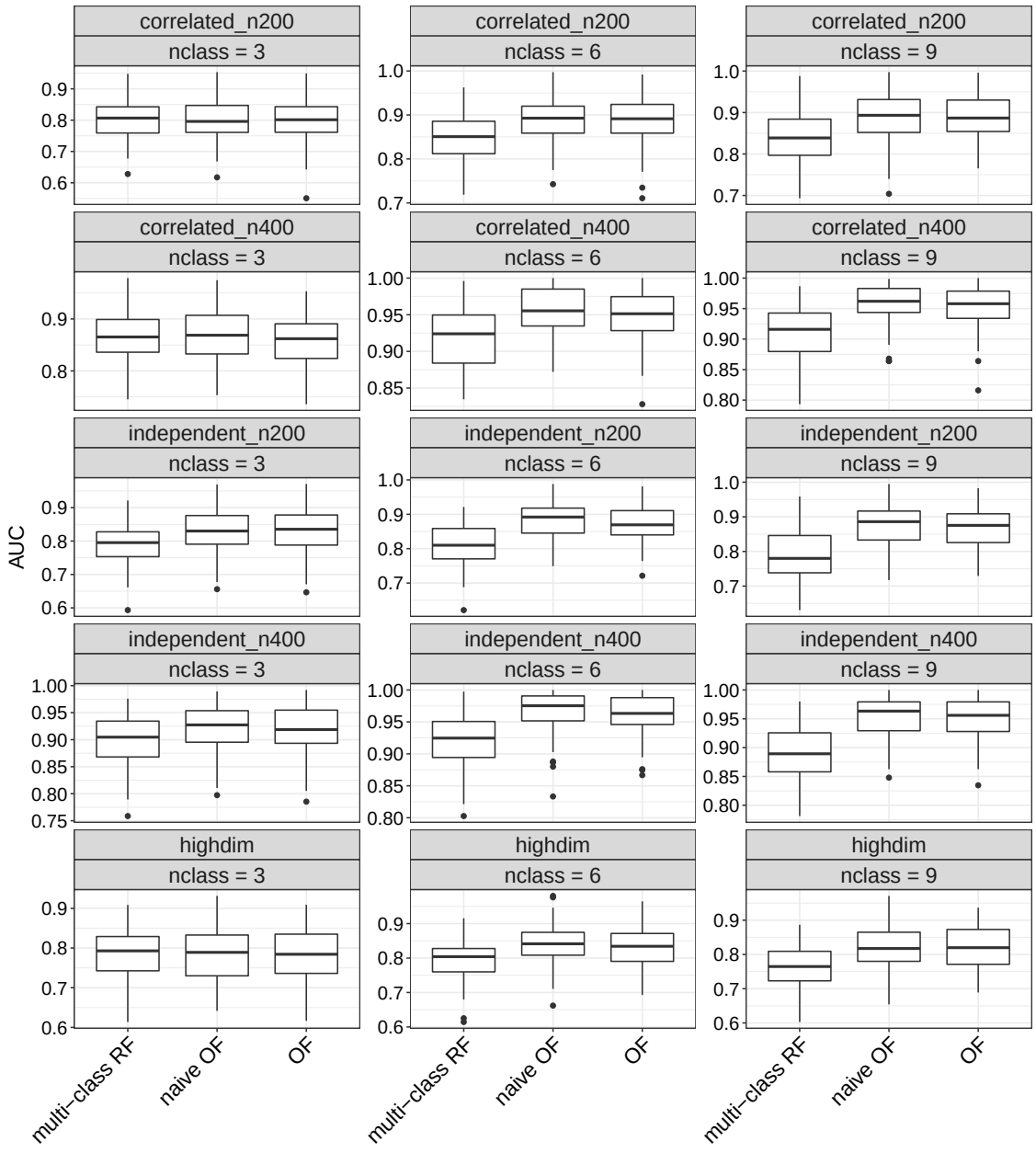


Supplementary Figure 5: Values of quadratically weighted Kappa for each simulation setting with random class widths and each of the three methods considered. Each boxplot shows the values obtained on the corresponding test dataset for each of the 100 simulation iterations.

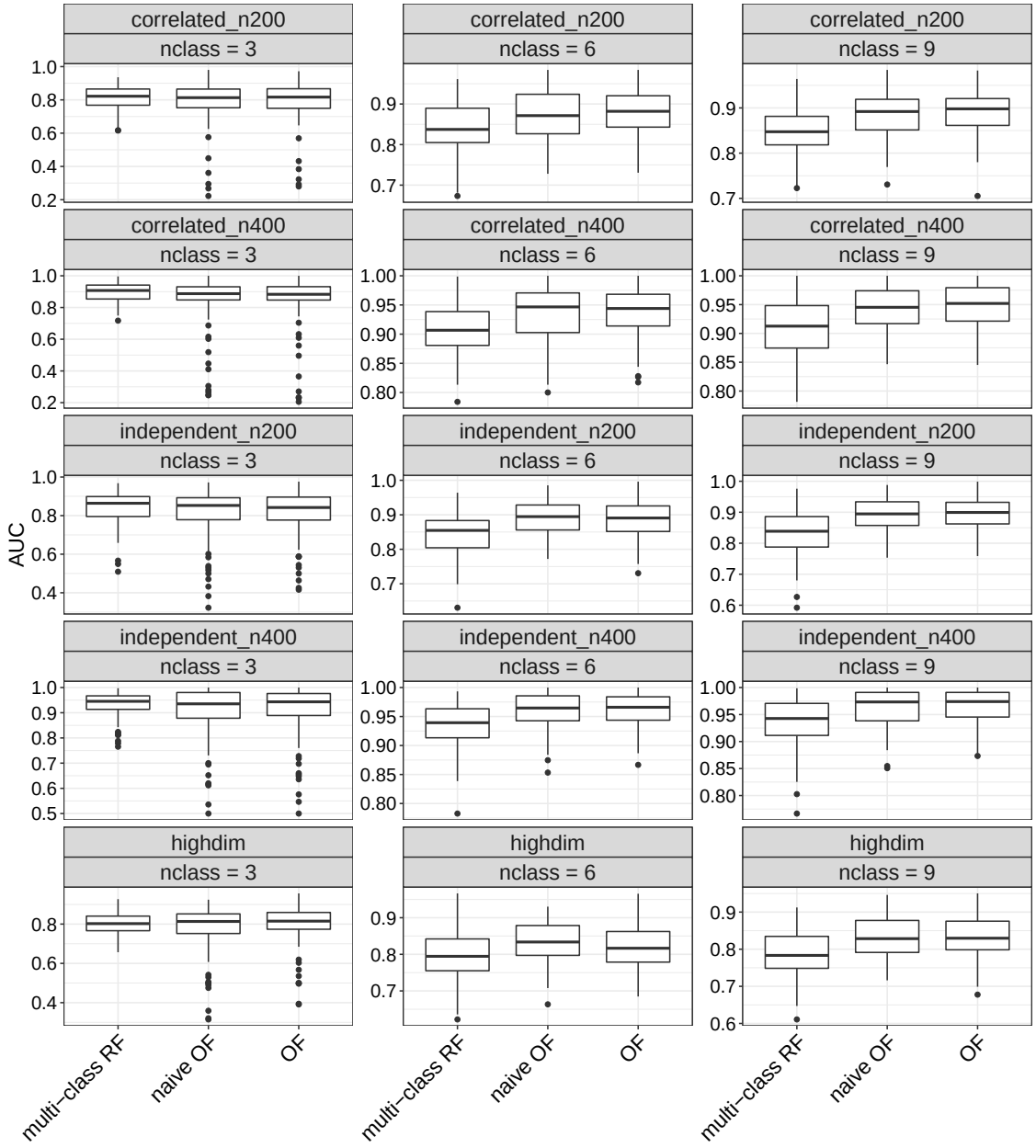


Supplementary Figure 6: Values of Cohen's Kappa for each simulation setting with random class widths and each of the three methods considered. Each boxplot shows the values obtained on the corresponding test dataset for each of the 100 simulation iterations.

D Supplementary Figures: simulation - AUC values obtained for variable importance measures



Supplementary Figure 7: AUC values associated with the VIMs for each simulation setting with equal class widths and each of the three methods considered. Each boxplot shows the values obtained for the 100 simulation iterations.



Supplementary Figure 8: AUC values associated with the VIMs for each simulation setting with random class widths and each of the three methods considered. Each boxplot shows the values obtained for the 100 simulation iterations.

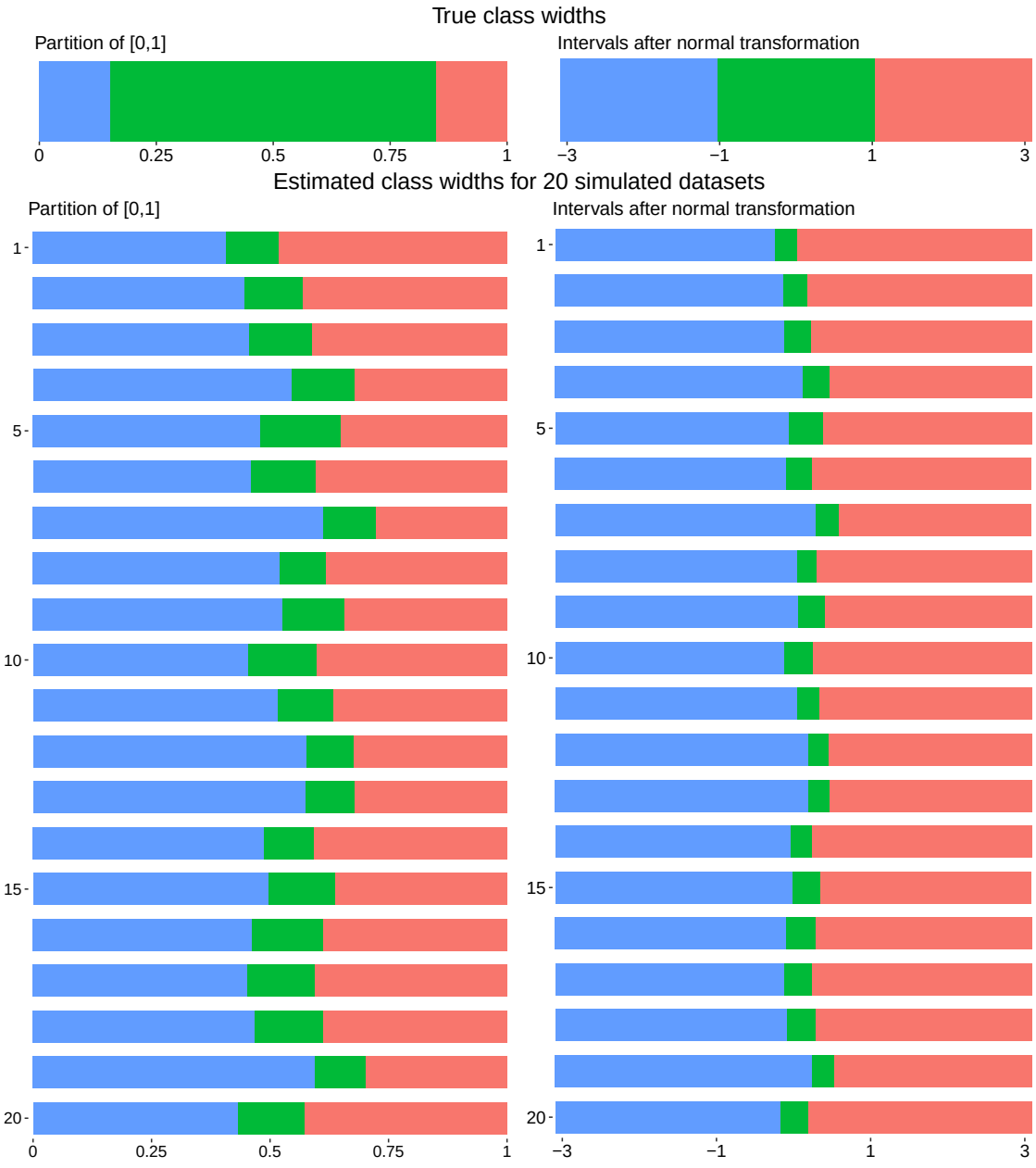
E Estimation of class widths

In the following only the settings with the covariates and sample size scenarios "correlated_n400", "independent_n400", and "highdim" are considered for the following reasons: The remaining covariates and sample size scenarios "correlated_n200" and "independent_n200" distinguish themselves from "correlated_n400" and "independent_n400" merely by a smaller number of training observations. Using the scenarios with higher number of training observations, more reliable results can be expected. Moreover, considering less settings makes the interpretation of the results easier.

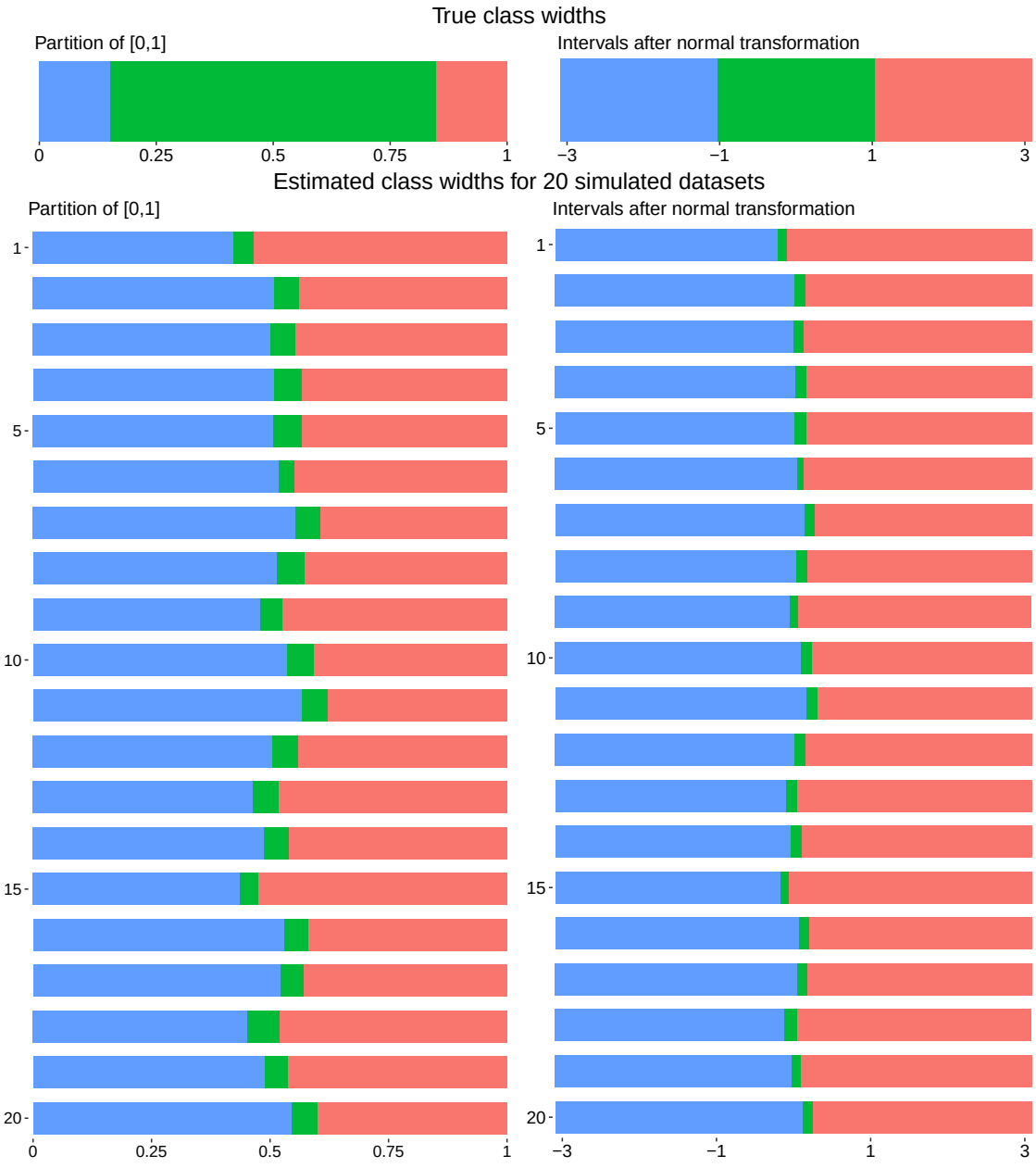
E.1 Equal class widths

In terms of prediction performance, for the settings with equal class widths OF outperformed naive OF to a considerably stronger degree than for the settings with random class widths (see section 3.2.2 of the main paper). Supplementary Figures 9 to 17 show for 20 simulated datasets from each of the considered settings the estimated partitions of $[0, 1]$ resulting from the OF algorithm, before and after transforming the interval borders by the quantile function of the standard normal distribution ("ϕ^{-1}-transforming"). For comparison, the true partitions of $[0, 1]$ before and after ϕ^{-1}-transforming are shown as well (see again section 3.2.1 of the main paper for the simulation design). In all cases, we clearly observe that the widths of the classes close to the center of the class value ranges that are represented by many observations in the dataset are associated with very small estimated widths. Correspondingly, the class widths of the classes close to the margins of the class value ranges that are represented by fewer observations are estimated much larger. Exceptions of this tendency of larger estimated class widths for classes close to the margins of the class value ranges are classes that are both very close to the margin and are represented by very few observations.

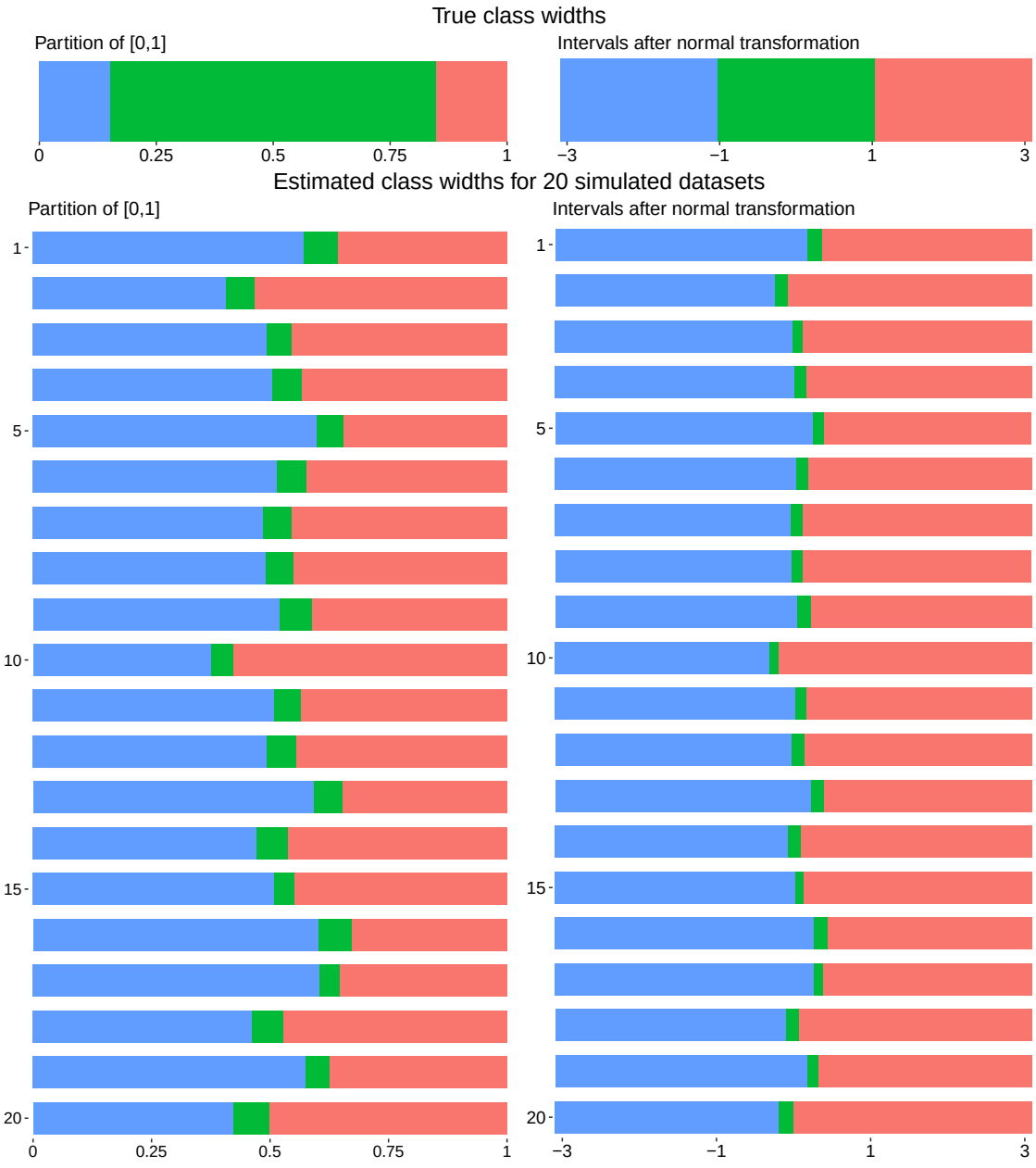
In the next subsection, in cases in which there is no chance of confusion, the true partitions of $[0, 1]$ underlying the simulated datasets will often be denoted shortly as "(true) partitions" and the estimated partitions $[0, 1]$ from the OF algorithm as "estimated partitions". In the descriptions only the versions of these partitions before ϕ^{-1}-transforming will be considered, not the ϕ^{-1}-transformed versions. This choice was made because first, the partitions of $[0, 1]$ are bounded, second, the conclusions drawn from considering both version are practically identical due to the one-to-one relationship between the partitions before and after ϕ^{-1}-transforming, and third, for the sake of clarity.



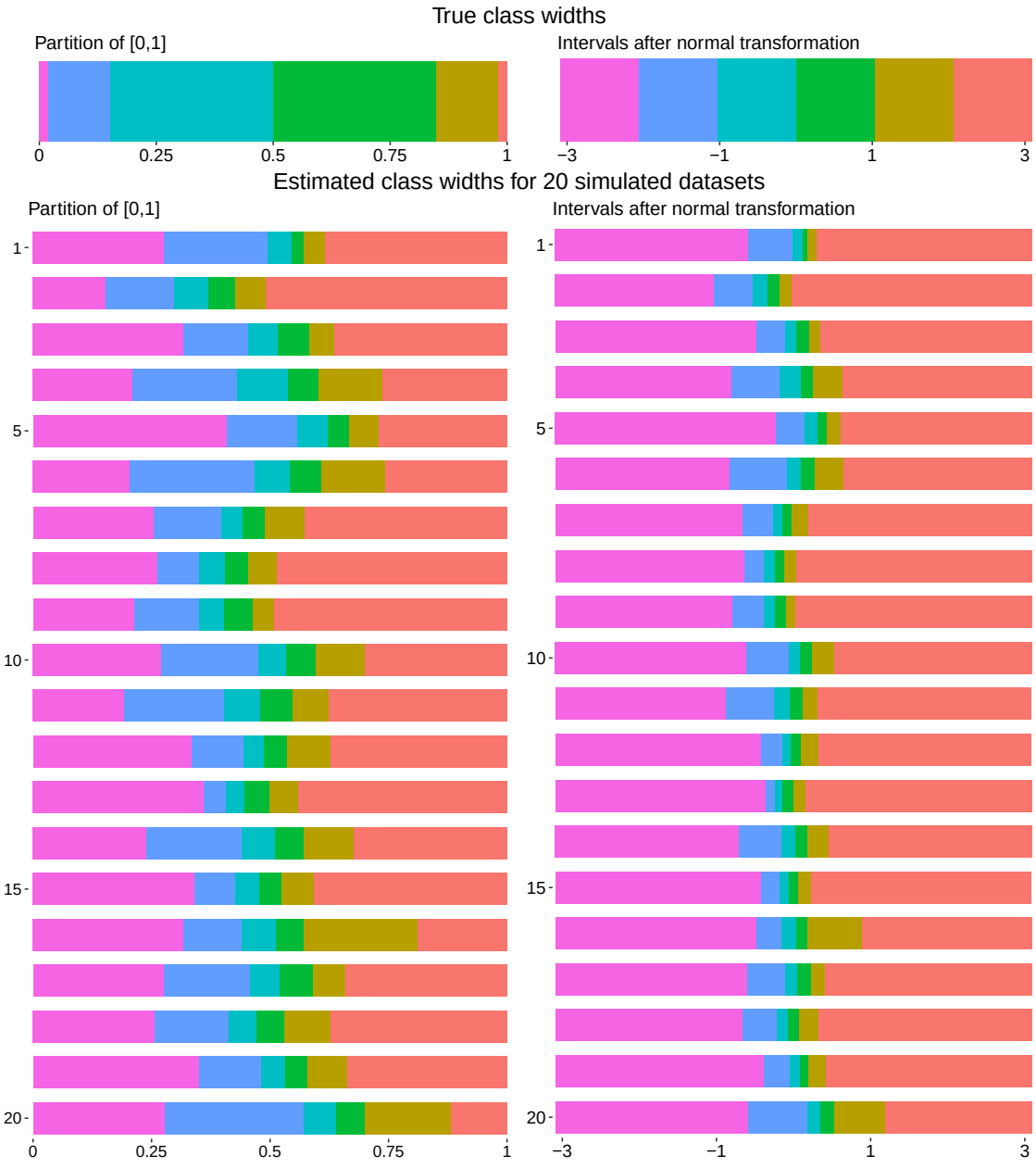
Supplementary Figure 9: True and estimated class widths for the setting with equal class widths, covariates and sample size scenario "correlated_n400", and "nclass = 3". Upper panels: true partition of $[0, 1]$ before (left panel) and after (right panel) transforming the interval borders by the quantile function of the standard normal distribution (ϕ^{-1}-transforming); lower panels: estimated partitions of $[0, 1]$ for 20 simulated datasets before (left panel) and after ϕ^{-1}-transforming (right panel).



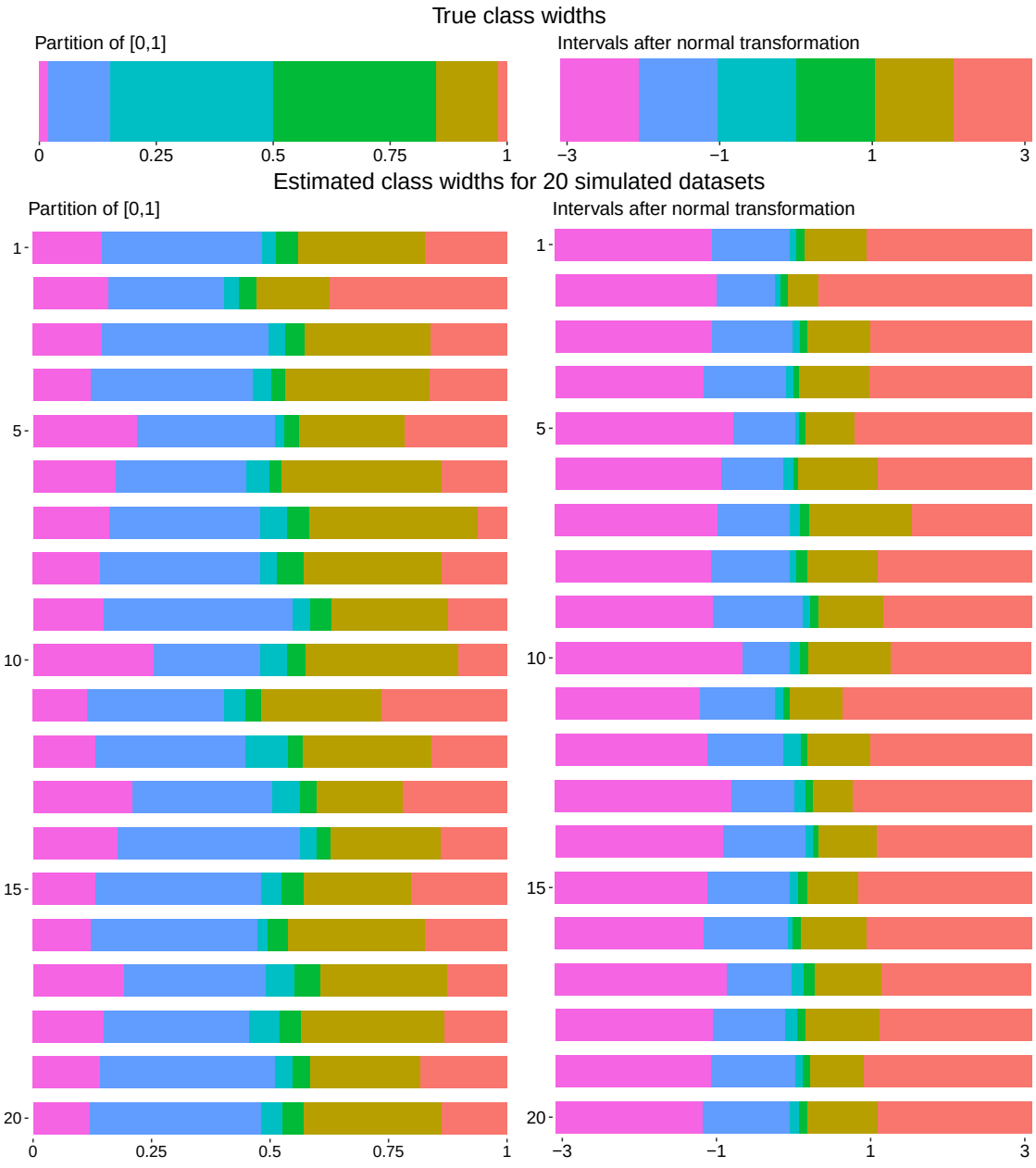
Supplementary Figure 10: True and estimated class widths for the setting with equal class widths, covariates and sample size scenario "independent_n400", and "nclass = 3". Upper panels: true partition of $[0, 1]$ before (left panel) and after (right panel) transforming the interval borders by the quantile function of the standard normal distribution (ϕ^{-1}-transforming); lower panels: estimated partitions of $[0, 1]$ for 20 simulated datasets before (left panel) and after ϕ^{-1}-transforming (right panel).



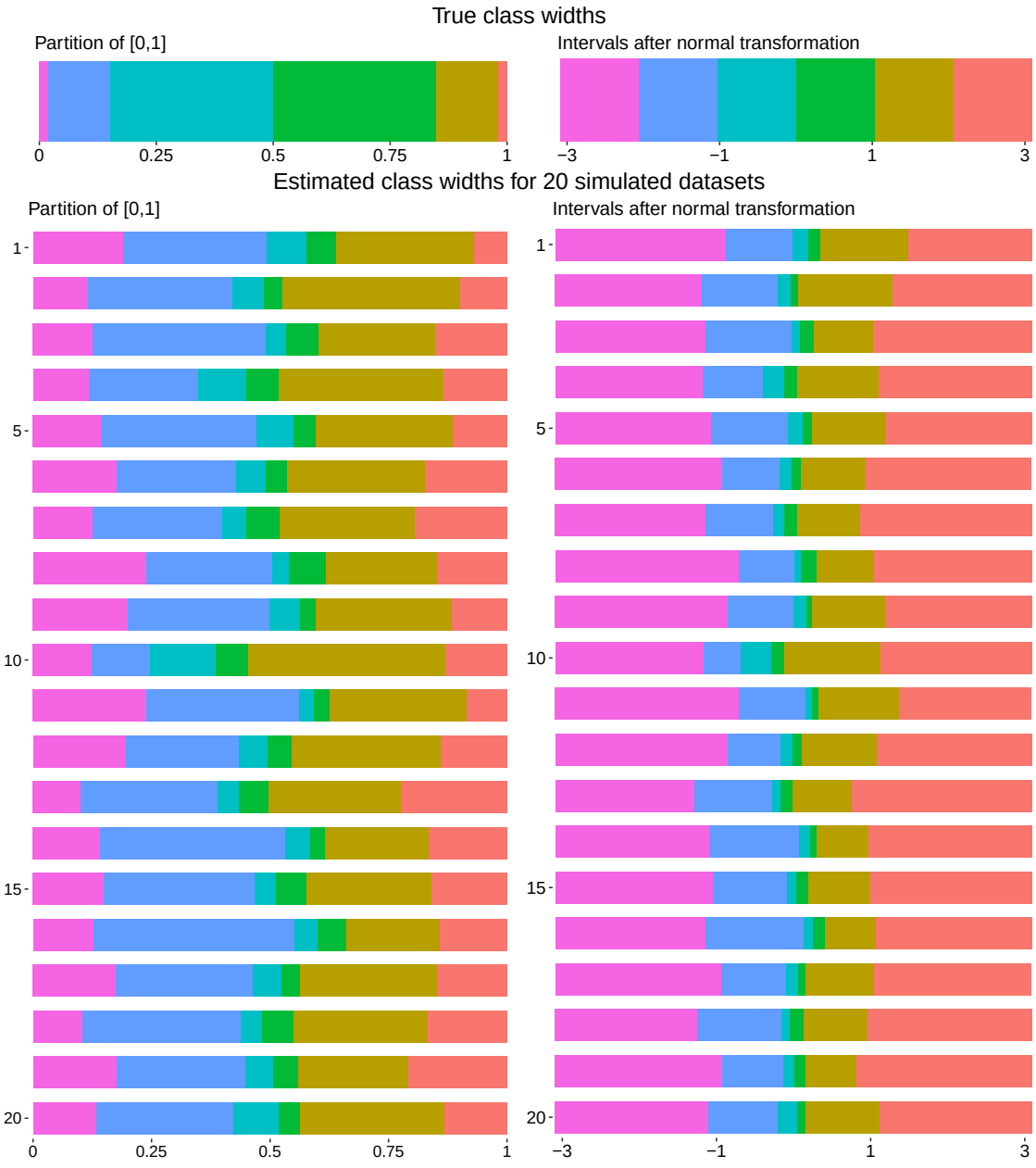
Supplementary Figure 11: True and estimated class widths for the setting with equal class widths, covariates and sample size scenario "highdim", and "nclass = 3". Upper panels: true partition of $[0, 1]$ before (left panel) and after (right panel) transforming the interval borders by the quantile function of the standard normal distribution (ϕ^{-1}-transforming); lower panels: estimated partitions of $[0, 1]$ for 20 simulated datasets before (left panel) and after ϕ^{-1}-transforming (right panel).



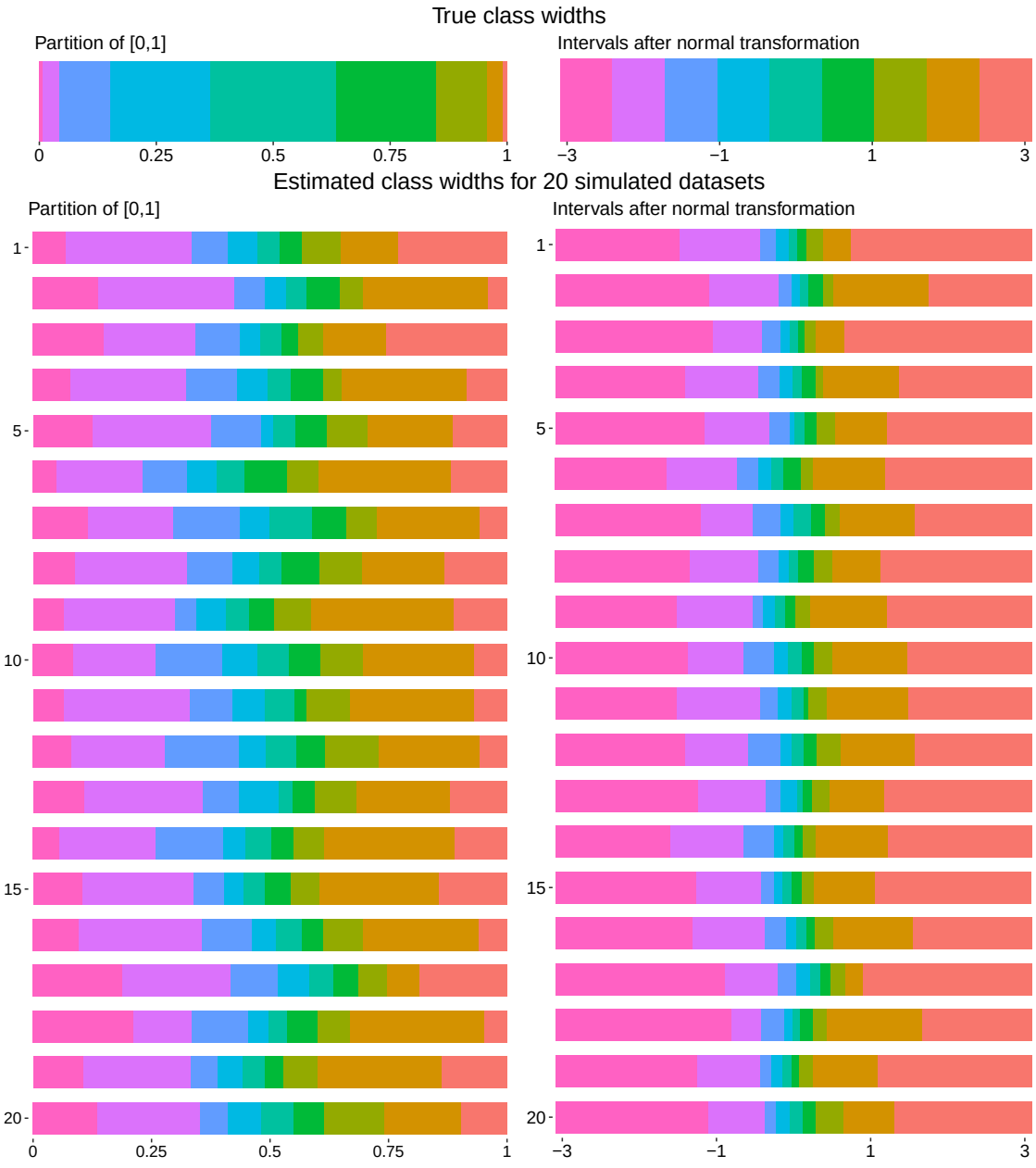
Supplementary Figure 12: True and estimated class widths for the setting with equal class widths, covariates and sample size scenario "correlated_n400", and "nclass = 6". Upper panels: true partition of $[0, 1]$ before (left panel) and after (right panel) transforming the interval borders by the quantile function of the standard normal distribution (ϕ^{-1}-transforming); lower panels: estimated partitions of $[0, 1]$ for 20 simulated datasets before (left panel) and after ϕ^{-1}-transforming (right panel).



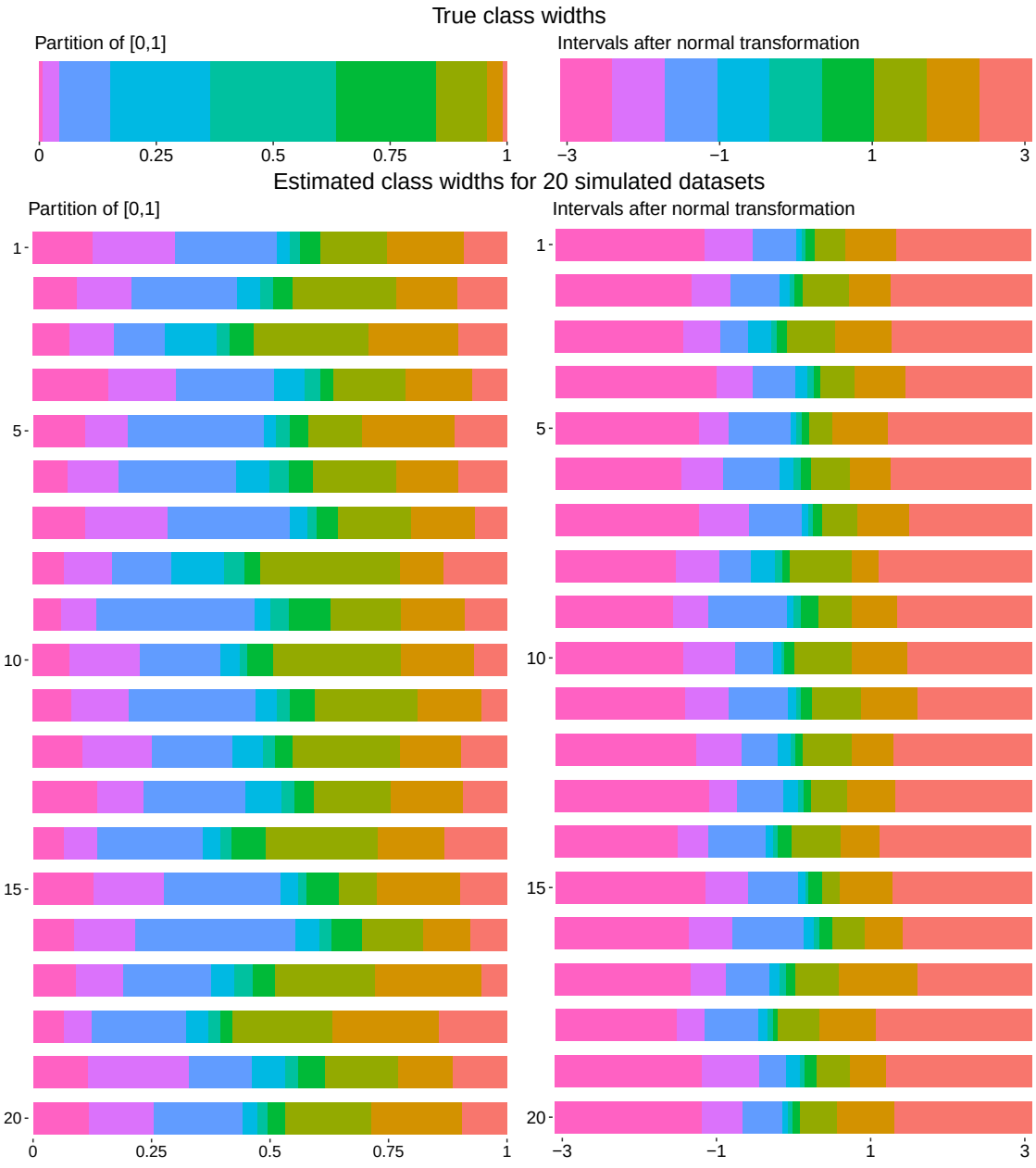
Supplementary Figure 13: True and estimated class widths for the setting with equal class widths, covariates and sample size scenario "independent_n400", and "nclass = 6". Upper panels: true partition of $[0, 1]$ before (left panel) and after (right panel) transforming the interval borders by the quantile function of the standard normal distribution (ϕ^{-1}-transforming); lower panels: estimated partitions of $[0, 1]$ for 20 simulated datasets before (left panel) and after ϕ^{-1}-transforming (right panel).



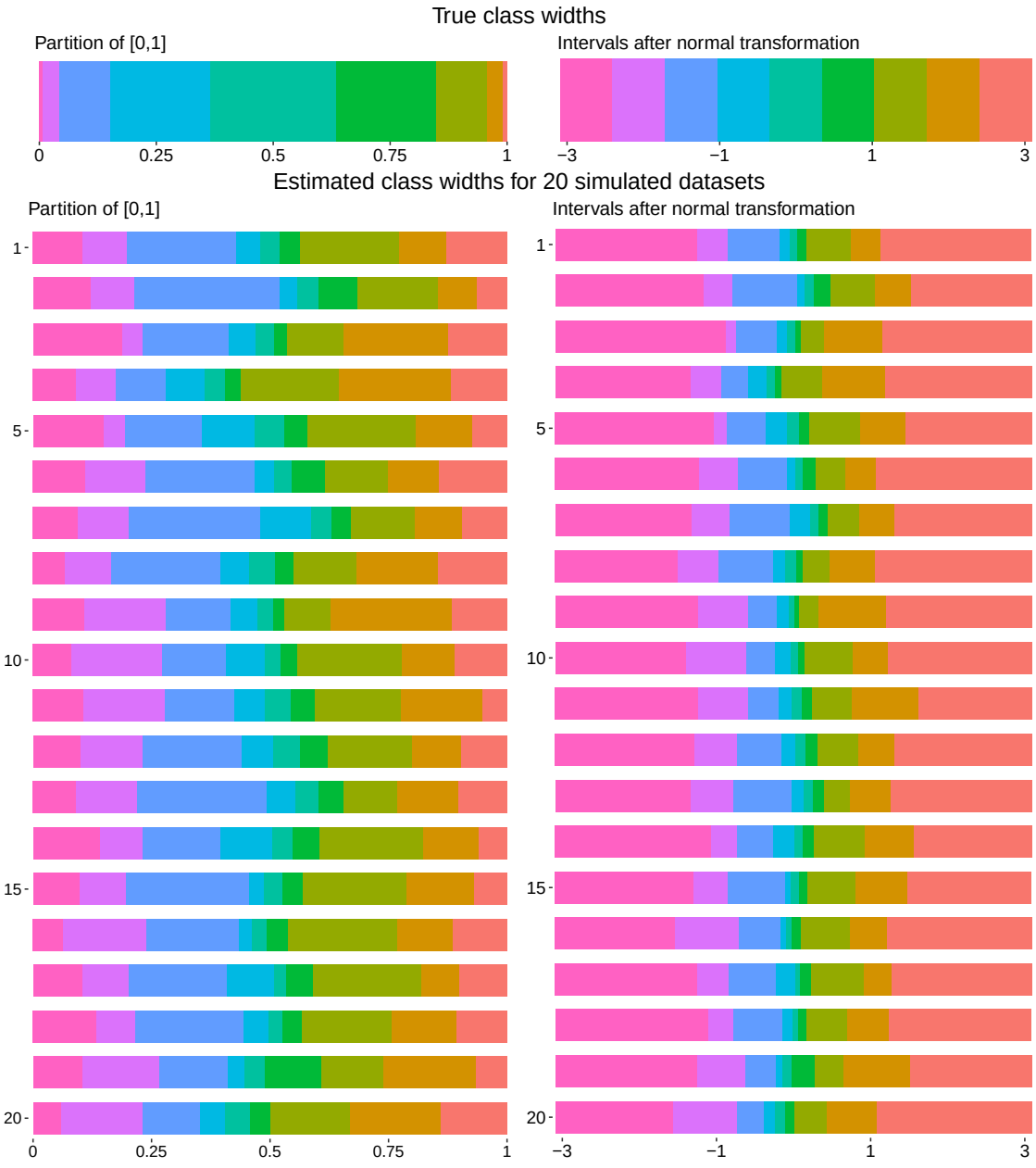
Supplementary Figure 14: True and estimated class widths for the setting with equal class widths, covariates and sample size scenario "highdim", and "nclass = 6". Upper panels: true partition of $[0, 1]$ before (left panel) and after (right panel) transforming the interval borders by the quantile function of the standard normal distribution (ϕ^{-1}-transforming); lower panels: estimated partitions of $[0, 1]$ for 20 simulated datasets before (left panel) and after ϕ^{-1}-transforming (right panel).



Supplementary Figure 15: True and estimated class widths for the setting with equal class widths, covariates and sample size scenario "correlated_n400", and "nclass = 9". Upper panels: true partition of $[0, 1]$ before (left panel) and after (right panel) transforming the interval borders by the quantile function of the standard normal distribution (ϕ^{-1}-transforming); lower panels: estimated partitions of $[0, 1]$ for 20 simulated datasets before (left panel) and after ϕ^{-1}-transforming (right panel).



Supplementary Figure 16: True and estimated class widths for the setting with equal class widths, covariates and sample size scenario "independent_n400", and "nclass = 9". Upper panels: true partition of $[0, 1]$ before (left panel) and after (right panel) transforming the interval borders by the quantile function of the standard normal distribution (ϕ^{-1}-transforming); lower panels: estimated partitions of $[0, 1]$ for 20 simulated datasets before (left panel) and after ϕ^{-1}-transforming (right panel).



Supplementary Figure 17: True and estimated class widths for the setting with equal class widths, covariates and sample size scenario "highdim", and "nclass = 9". Upper panels: true partition of $[0, 1]$ before (left panel) and after (right panel) transforming the interval borders by the quantile function of the standard normal distribution (ϕ^{-1}-transforming); lower panels: estimated partitions of $[0, 1]$ for 20 simulated datasets before (left panel) and after ϕ^{-1}-transforming (right panel).

E.2 Random class widths

Supplementary Figures 18 to 26 show for 10 simulated datasets the true and estimated partitions of $[0, 1]$ before and after ϕ^{-1}-transforming. Again we observe that the classes closer to the margins of the class value range tend to be associated with larger estimated class widths than the classes close to the middle of the class value range. There is again no apparent resemblance between the true and estimated partitions. In the special case of "nclass = 3", there is a strong negative correlation between the class widths of the true and estimated partitions for classes 1 and 3.

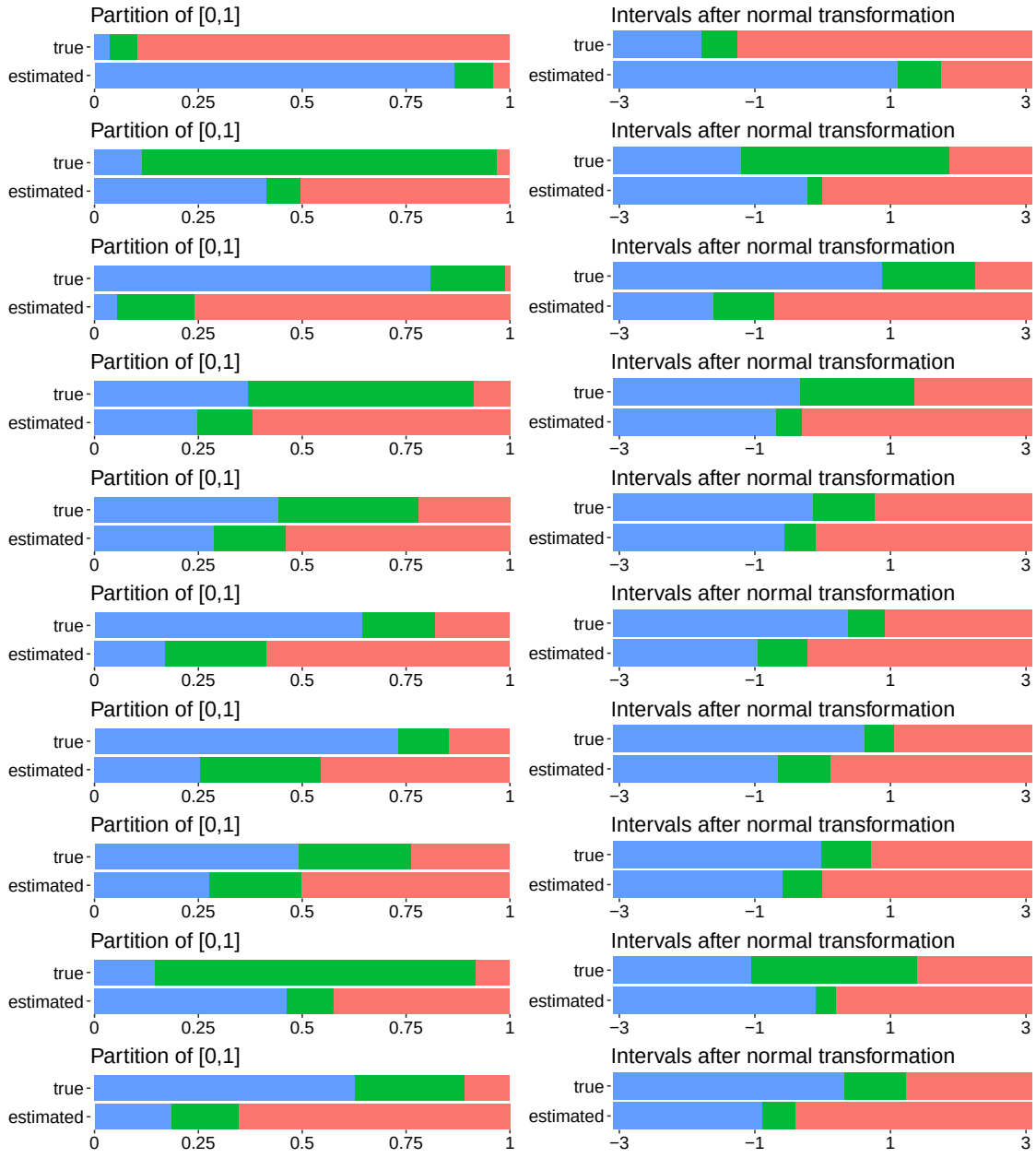
Supplementary Figures 27 to 29 show scatterplots of the relationships between true and estimated class widths.

For all three studied covariates and sample size scenarios we observe that for "nclass = 3" large estimated class widths can result only for small true class widths. For larger true class widths the corresponding estimated class widths are small. Moreover, the three plots suggest that, with some exceptions, the widths of the class in the middle are estimated very small independent on whether the true class widths are small or large.

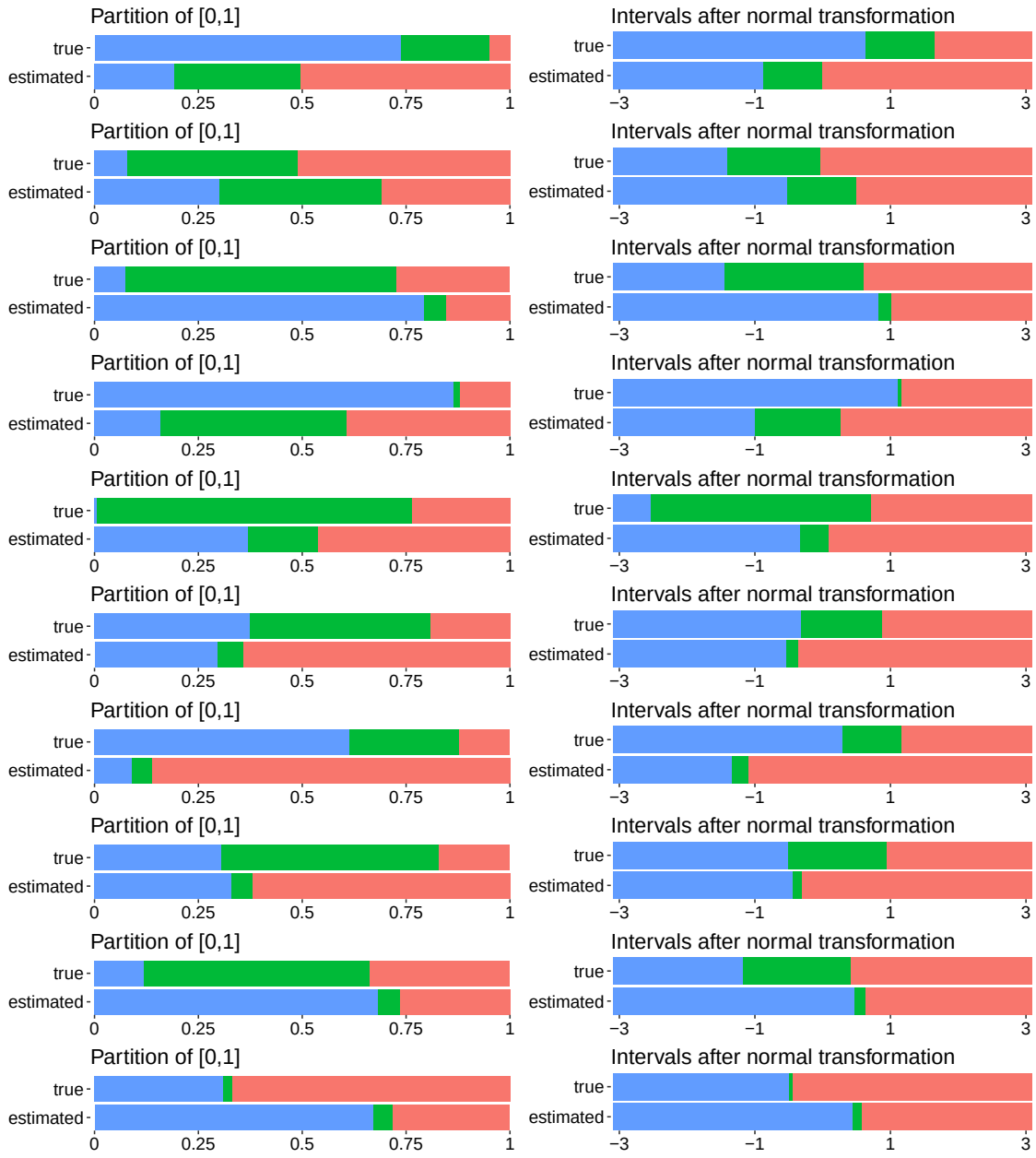
In the case of "nclass = 6" the picture differs between the three scenarios studied. For all three scenarios the widths of the classes on the margins of the class value range are estimated larger than the other classes. Moreover, in all three scenarios except for in the cases of the classes on the margins of the class value range there is no dependency of the estimated class widths on the true class widths. However, scenario "correlated_n400" differs from the scenarios "independent_n400' and "highdim" in the following two points: 1) For "correlated_n400" there is a strong negative correlation between the true and estimated class widths for the classes on the margins of the class value range (i.e., classes 1 and 6), whereas for the other two scenarios this correlation is positive; 2) For "correlated_n400" the widths of classes 2 to 5 are not estimated systematically different, whereas for the other two scenarios the classes closer to the middle of the class value ranges are estimated smaller. As stated in section 3.2.2 of the main paper, for scenario "correlated_n400" the dependency of the continuous response variable on the covariates is less refined than for scenario "independent_n400" with frequent very large or very small values of the linear predictor. This leads to a greater signal with respect to explaining the classes on the margins of the class value range. This better predictability of the low and high classes for scenario "correlated_n400" helps to explain why for this scenario there is such a strong relation between true and estimated class widths for these classes. While the coarseness of the signal for this scenario is associated with a better predictability of the classes on the margins of the class value range it also leads to a low discriminability between the other classes. The latter might explain why there are no systematic differences between the estimated class widths for these classes. Other than for the classes on the margins of the class value range, there might simply be no optimal class widths for particular classes that would lead to a better predictability of these classes. The fact that for scenario "highdim" we do not observe a similar picture as for scenario "correlated_n400" but instead a similar picture as for scenario "independent_n400" is probably explainable by the fact that the strong correlations between the influential variables in "highdim" are not as influential as in "correlated_n400" due to the high number of covariates in "highdim".

For the settings with "nclass = 9" there are no appreciably strong dependencies of the estimated class widths on the true class widths. However, also here the widths of the classes in the middle

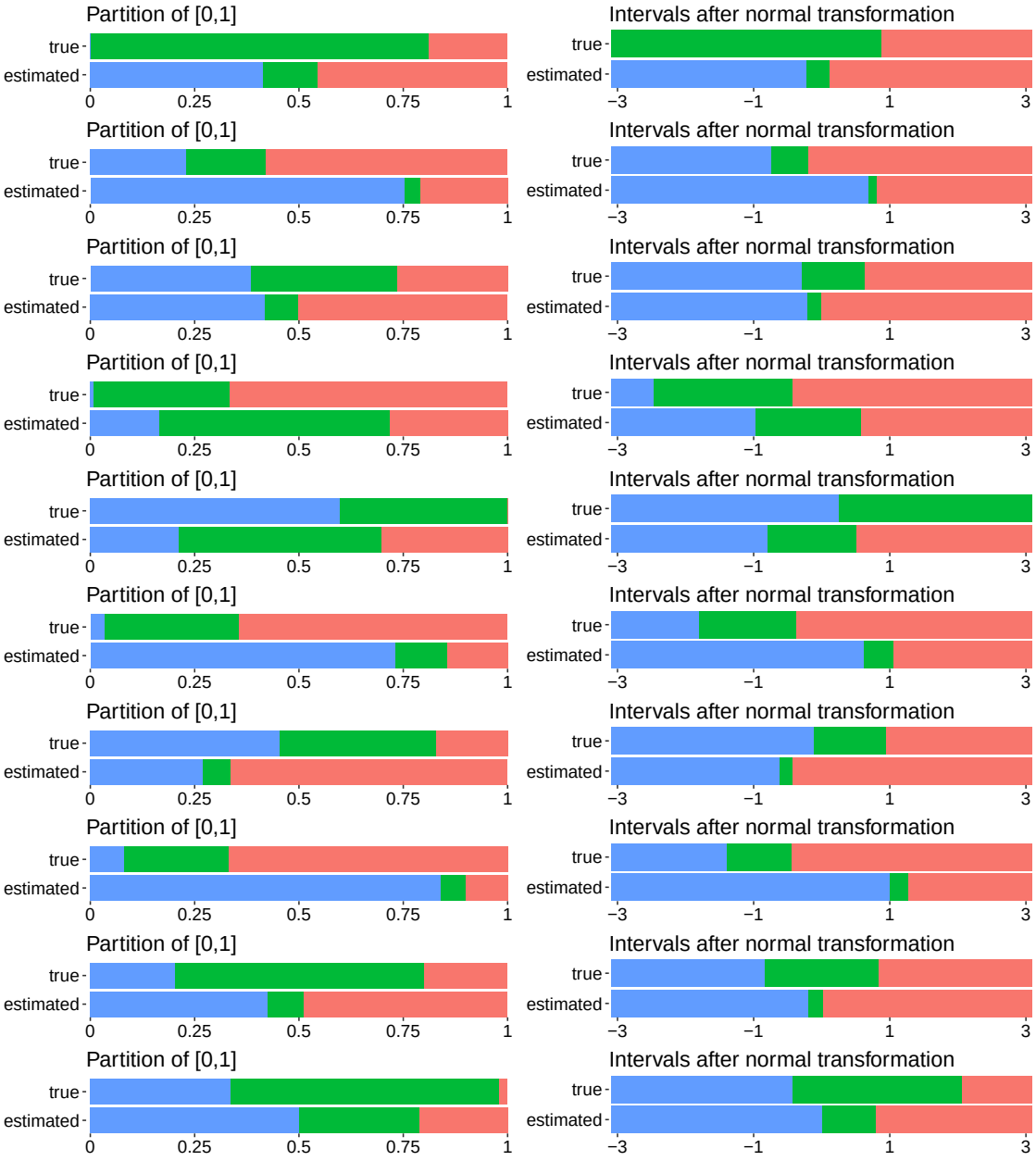
are estimated smaller than the widths of the low and high classes, where again for the scenario "correlated_n400" the widths of the classes in the middle do not differ systematically from each other.



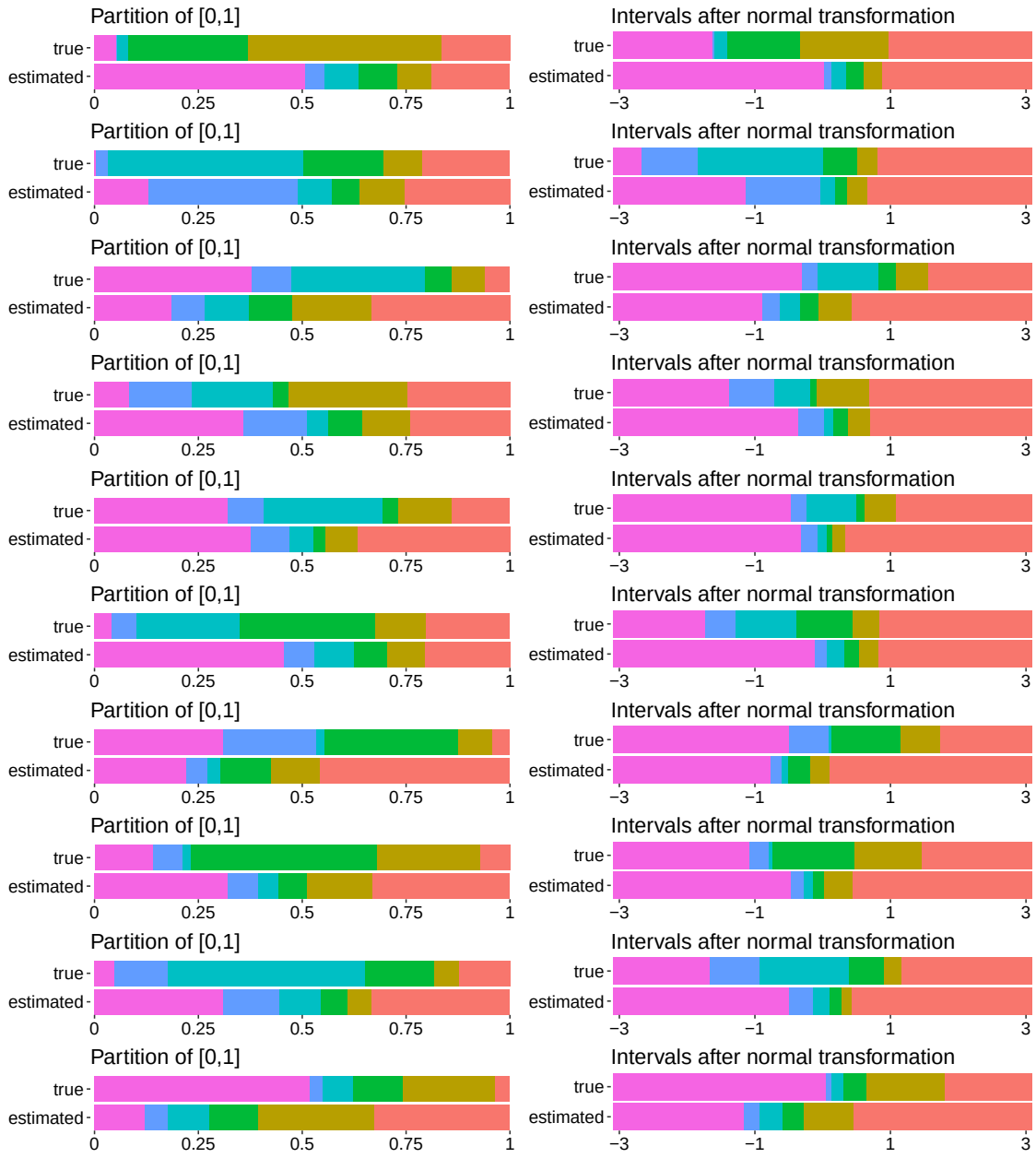
Supplementary Figure 18: True and estimated class widths for 10 simulated datasets for the setting with random class widths, covariates and sample size scenario "correlated_n400", and "nclass = 3". Each row shows for a particular dataset the true and estimated partition of $[0, 1]$ before (left panel) and after ϕ^{-1}-transforming (right panel).



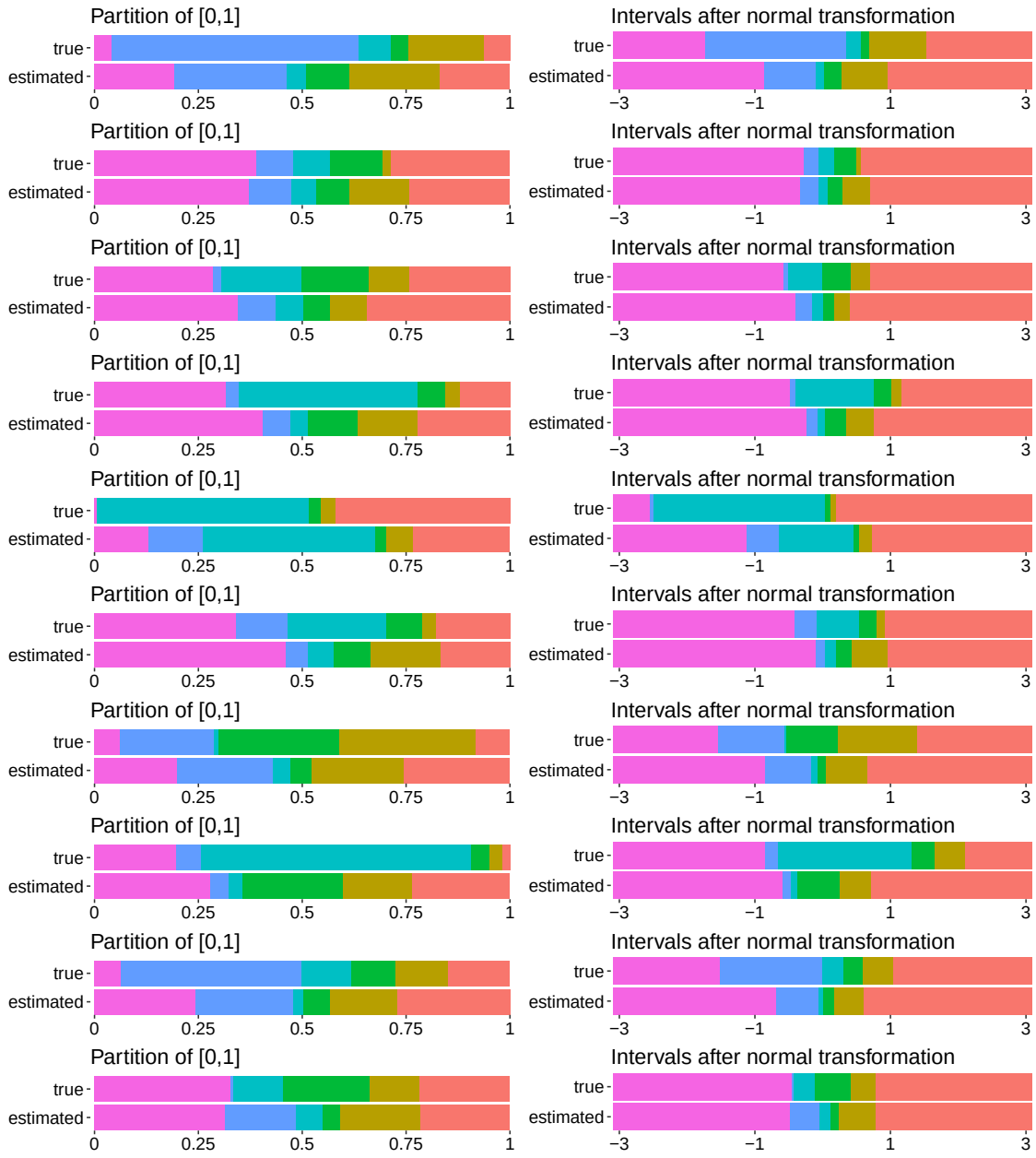
Supplementary Figure 19: True and estimated class widths for 10 simulated datasets for the setting with random class widths, covariates and sample size scenario "independent_n400", and "nclass = 3". Each row shows for a particular dataset the true and estimated partition of $[0, 1]$ before (left panel) and after ϕ^{-1}-transforming (right panel).



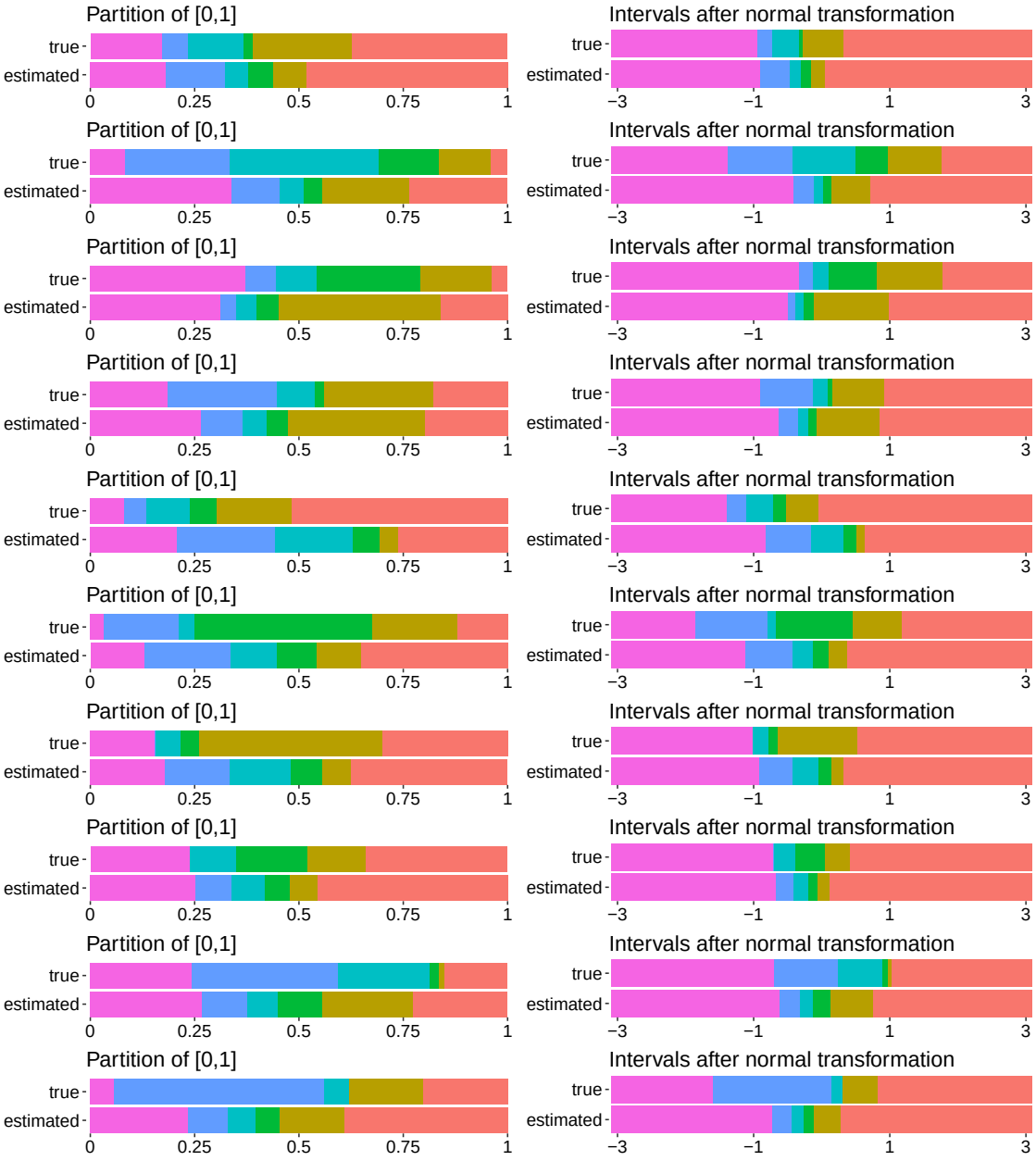
Supplementary Figure 20: True and estimated class widths for 10 simulated datasets for the setting with random class widths, covariates and sample size scenario "highdim", and "nclass = 3". Each row shows for a particular dataset the true and estimated partition of $[0, 1]$ before (left panel) and after ϕ^{-1}-transforming (right panel).



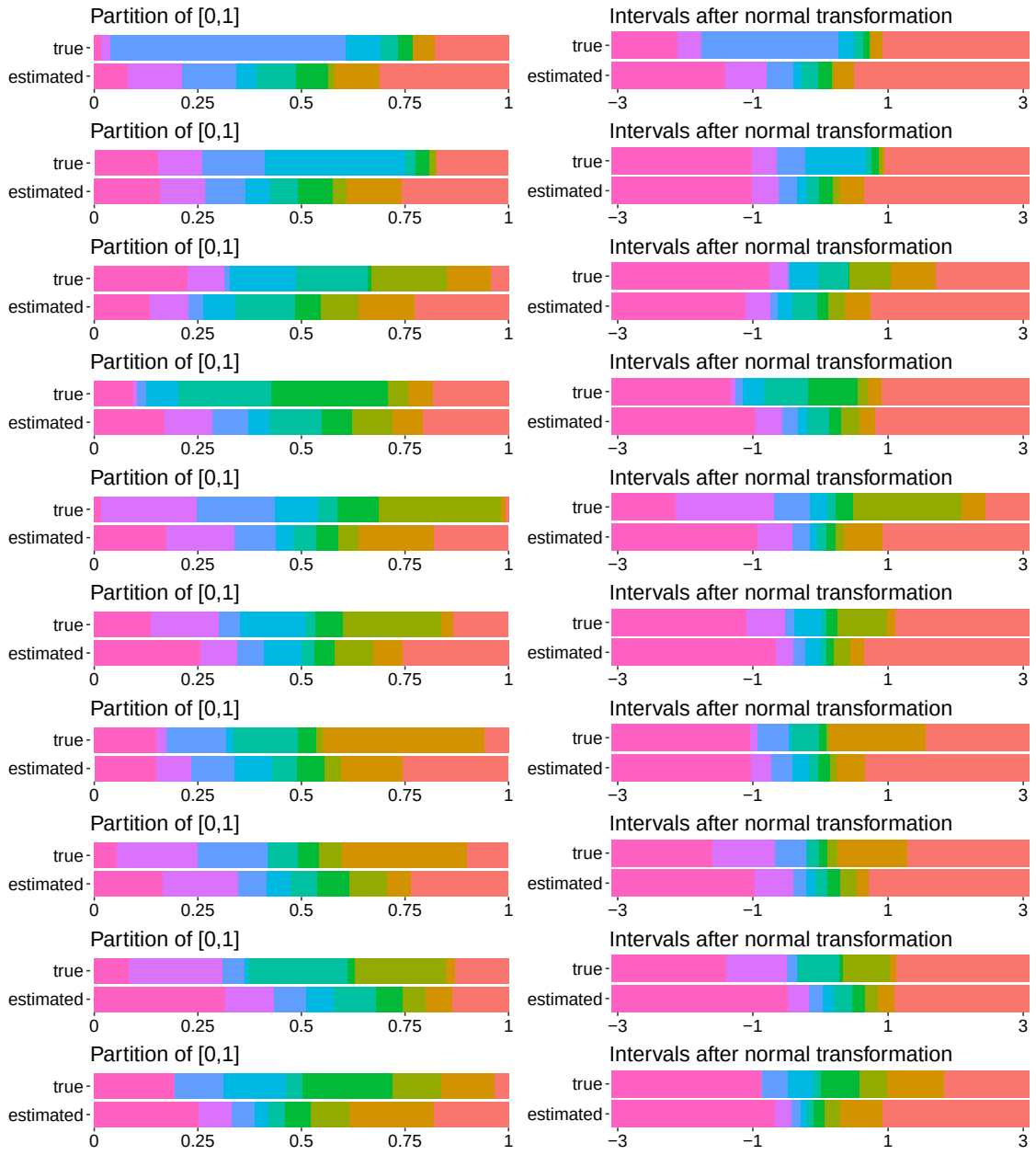
Supplementary Figure 21: True and estimated class widths for 10 simulated datasets for the setting with random class widths, covariates and sample size scenario "correlated_n400", and "nclass = 6". Each row shows for a particular dataset the true and estimated partition of $[0, 1]$ before (left panel) and after ϕ^{-1}-transforming (right panel).



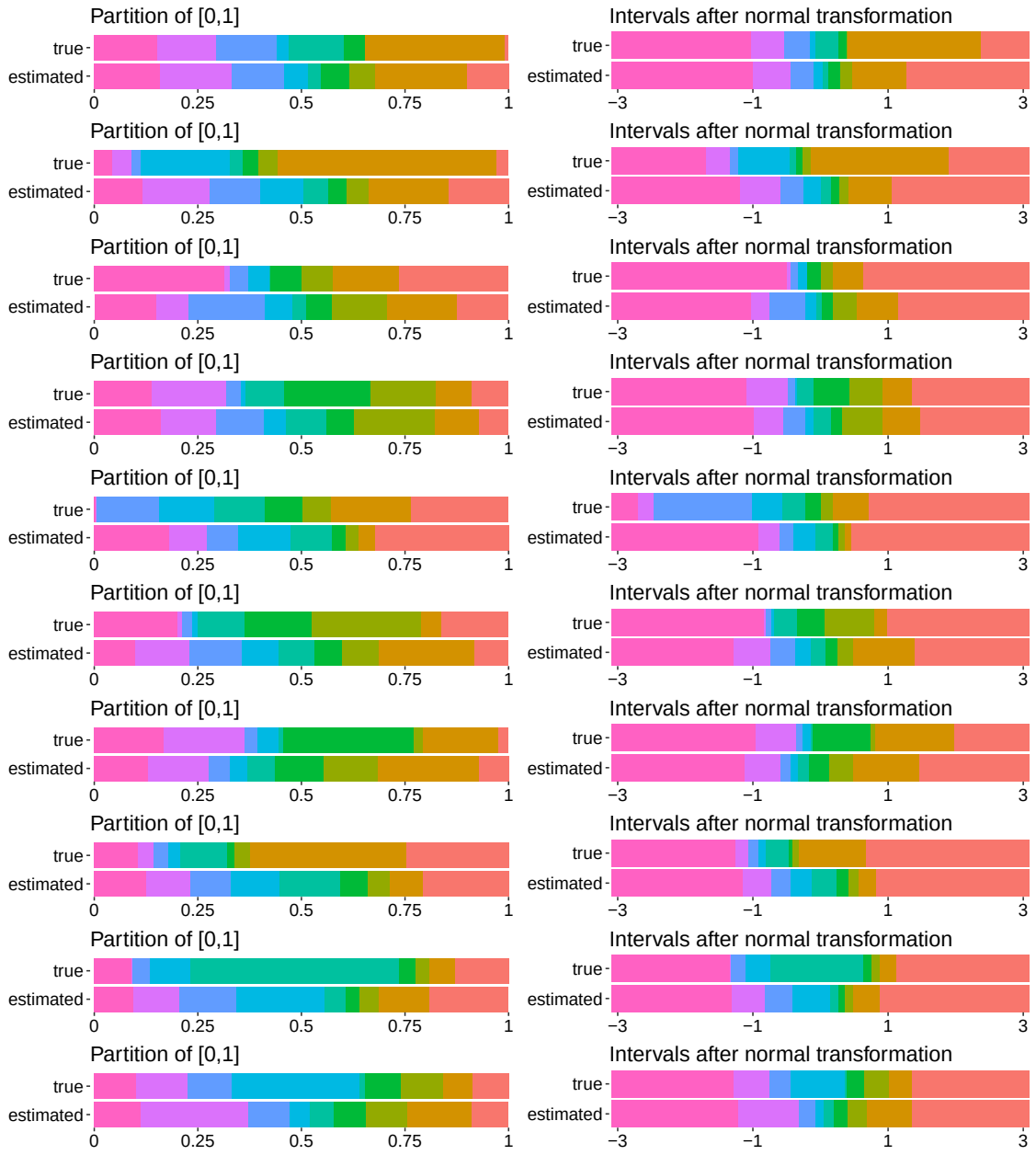
Supplementary Figure 22: True and estimated class widths for 10 simulated datasets for the setting with random class widths, covariates and sample size scenario "independent_n400", and "nclass = 6". Each row shows for a particular dataset the true and estimated partition of $[0, 1]$ before (left panel) and after ϕ^{-1}-transforming (right panel).



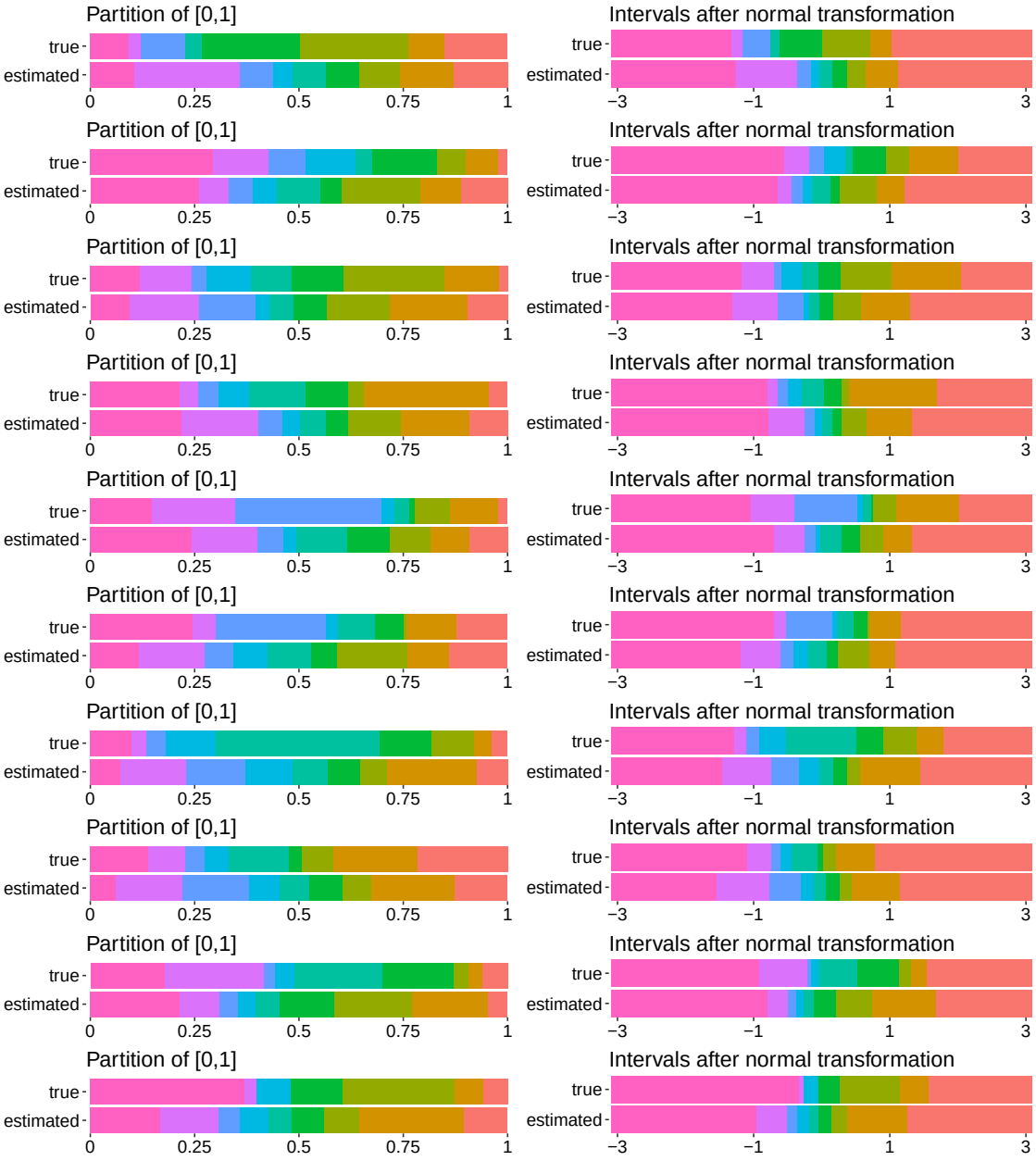
Supplementary Figure 23: True and estimated class widths for 10 simulated datasets for the setting with random class widths, covariates and sample size scenario "highdim", and "nclass = 6". Each row shows for a particular dataset the true and estimated partition of $[0, 1]$ before (left panel) and after ϕ^{-1}-transforming (right panel).



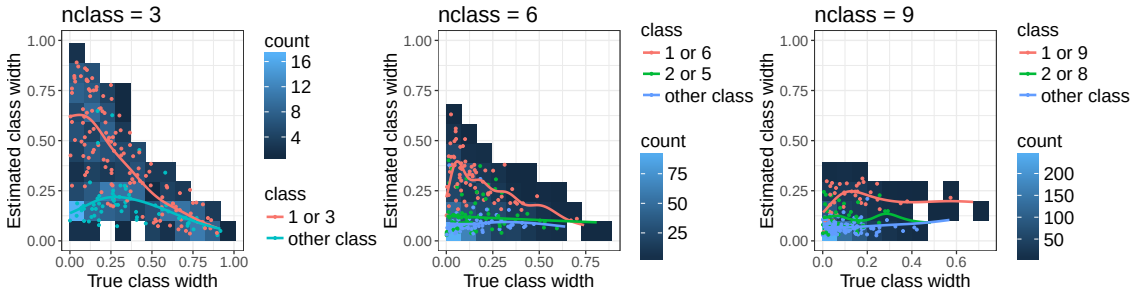
Supplementary Figure 24: True and estimated class widths for 10 simulated datasets for the setting with random class widths, covariates and sample size scenario "correlated_n400", and "nclass = 9". Each row shows for a particular dataset the true and estimated partition of $[0, 1]$ before (left panel) and after ϕ^{-1}-transforming (right panel).



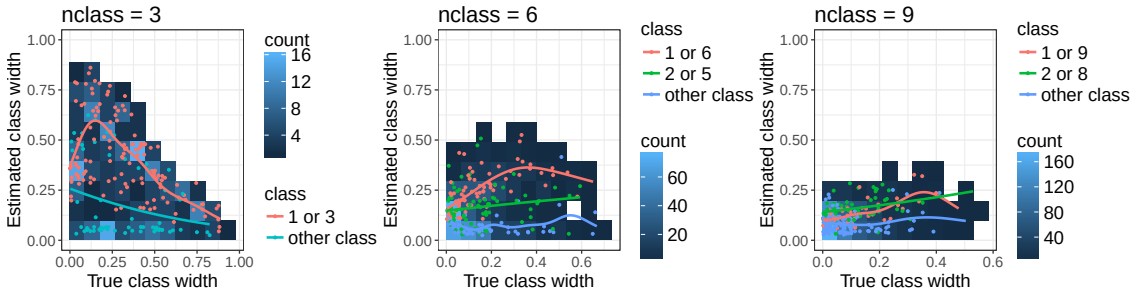
Supplementary Figure 25: True and estimated class widths for 10 simulated datasets for the setting with random class widths, covariates and sample size scenario "independent_n400", and "nclass = 9". Each row shows for a particular dataset the true and estimated partition of $[0, 1]$ before (left panel) and after ϕ^{-1}-transforming (right panel).



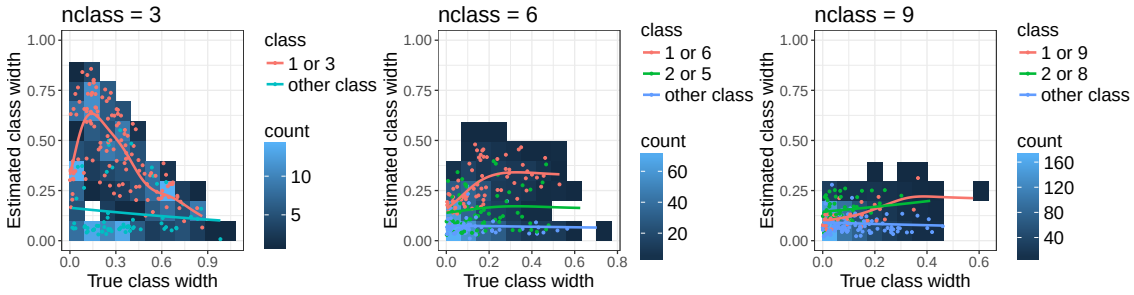
Supplementary Figure 26: True and estimated class widths for 10 simulated datasets for the setting with random class widths, covariates and sample size scenario "highdim", and "nclass = 9". Each row shows for a particular dataset the true and estimated partition of $[0, 1]$ before (left panel) and after ϕ^{-1}-transforming (right panel).



Supplementary Figure 27: Estimated versus true class widths for settings with random class widths and covariates and sample size scenario "correlated_n400". The lines show fits of quasi-binomial regression using penalized thin plate regression splines. The points show the pairs of true and corresponding estimated class widths, where in each case 200 pairs from all pairs available were subsampled for reasons of clarity.



Supplementary Figure 28: Estimated versus true class widths for settings with random class widths and covariates and sample size scenario "independent_n400". The lines show fits of quasi-binomial regression using penalized thin plate regression splines. The points show the pairs of true and corresponding estimated class widths, where in each case 200 pairs from all pairs available were subsampled for reasons of clarity.



Supplementary Figure 29: Estimated versus true class widths for settings with random class widths and covariates and sample size scenario "highdim". The lines show fits of quasi-binomial regression using penalized thin plate regression splines. The points show the pairs of true and corresponding estimated class widths, where in each case 200 pairs from all pairs available were subsampled for reasons of clarity.

F Influence of the class distribution on the performance of OF

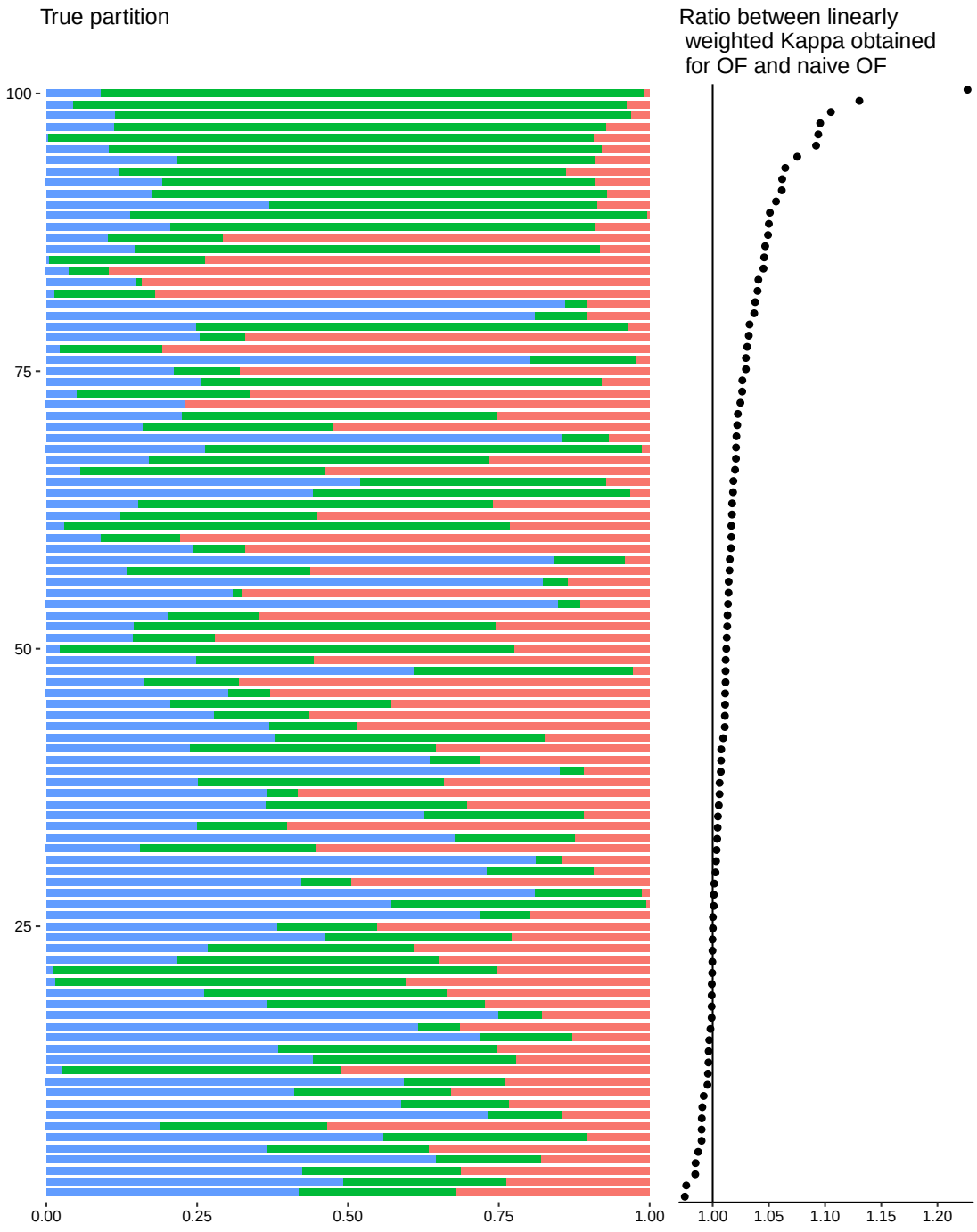
Again only the covariates and sample size scenarios "correlated_n400", "independent_n400", and "highdim" will be considered. Moreover, all those simulated datasets will be excluded for which OF or naive OF or both of the latter featured a linearly weighted Kappa value smaller than 0.1. This choice was made for two reasons: 1) to exclude pathological partitions, for example partitions for which some class widths are extremely small or for which one class width is very large; 2) to avoid outlying values of the ratio between the linearly weighted Kappa values obtained for OF and naive OF (see below) that would result when the smaller value of the pair of values is by chance much smaller than the higher value.

Supplementary Figures 30 to 38 show for each setting and each considered simulated dataset the true class distribution together with the ratio between the value of the linearly weighted Kappa obtained for OF and the corresponding value obtained for naive OF.

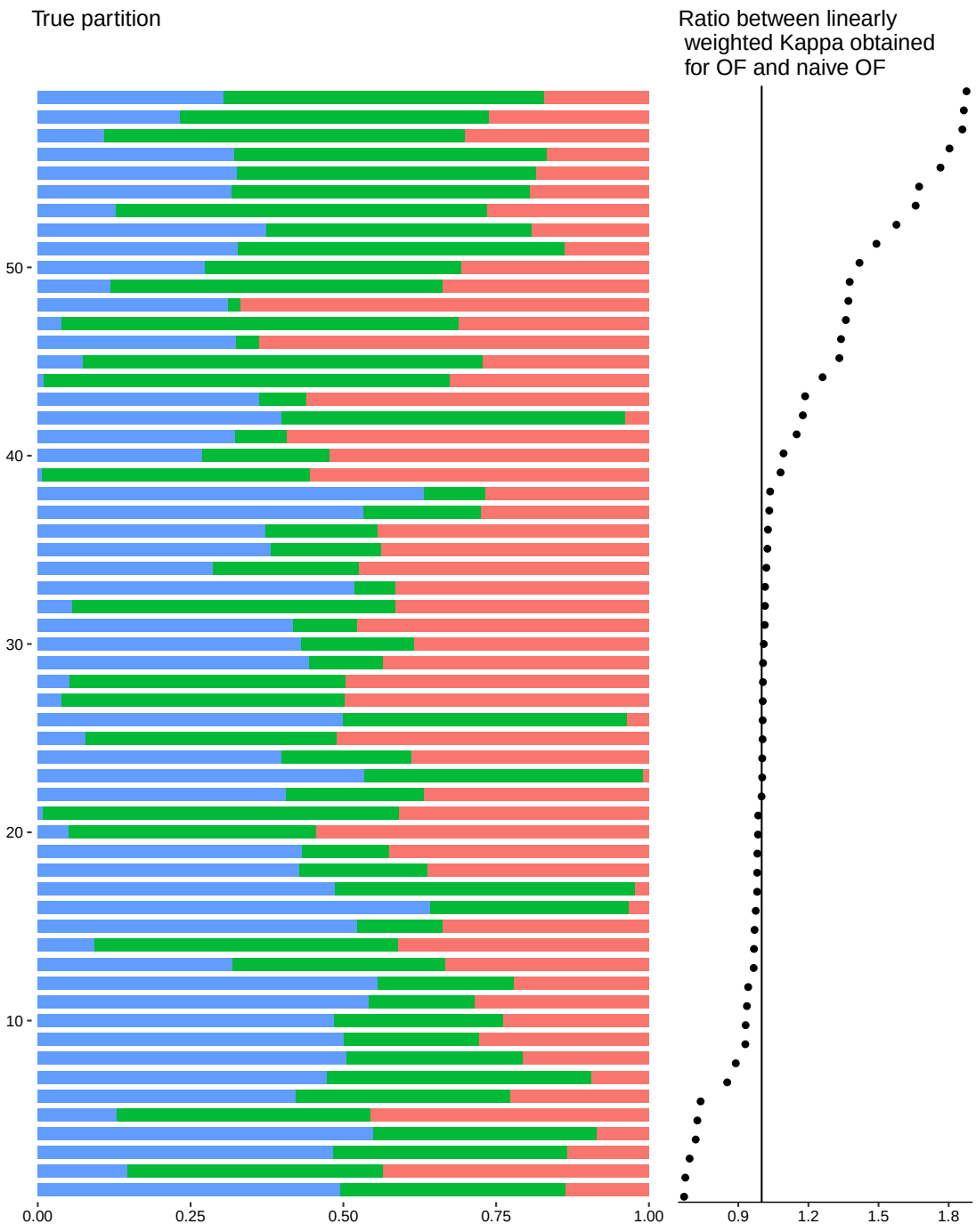
For the setting "correlated_n400" with "nclass = 3" we clearly observe that for those class distributions for which there is the greatest improvement of OF over naive OF, the width of the middle class (i.e., class 2) is much larger than the widths of classes 1 and 3. These class distributions, moreover, tend to be symmetric in the sense that the widths of class 1 and class 3 tend to be approximately equally wide. In the cases of the class distributions for which OF is slightly worse than naive OF the middle class features a comparably small width, where at least one of the classes 1 and 3 features a larger or much larger width. For the setting "independent_n400" with "nclass = 3" we make similar observations as for "correlated_n400". The corresponding effects, however, appear less strong, because for this scenario there are no class distributions for which the middle class features a very large width. This is because for this scenario the simulated datasets associated with such class distributions did not meet the filtering criterion on the minimum required classification performance described above. For the setting "highdim" with "nclass = 3" again similar observations are made. Nevertheless, in the case of this setting for some of the datasets with considerably better performance of OF than naive OF, the widths of the middle class are narrow.

In comparison to the settings with "nclass = 3" where the small number of classes made the interpretation of the results very easy, for "nclass = 6" it is less easy to recognize features of the class distributions associated with a particularly strong performance of OF. Nevertheless, the following tendencies can be identified easily studying the corresponding figures (Supplementary Figures 33 to 35): 1) Class distributions associated with a particularly well performance of OF in comparison to that of naive OF often feature small widths of the low and high classes and large widths of one or several classes around the center of the partitions; 2) Conversely, in cases for which OF is inferior to or at least not superior over naive OF, the corresponding class distributions tend to feature large widths of the low and high classes and small widths of the classes in the middle.

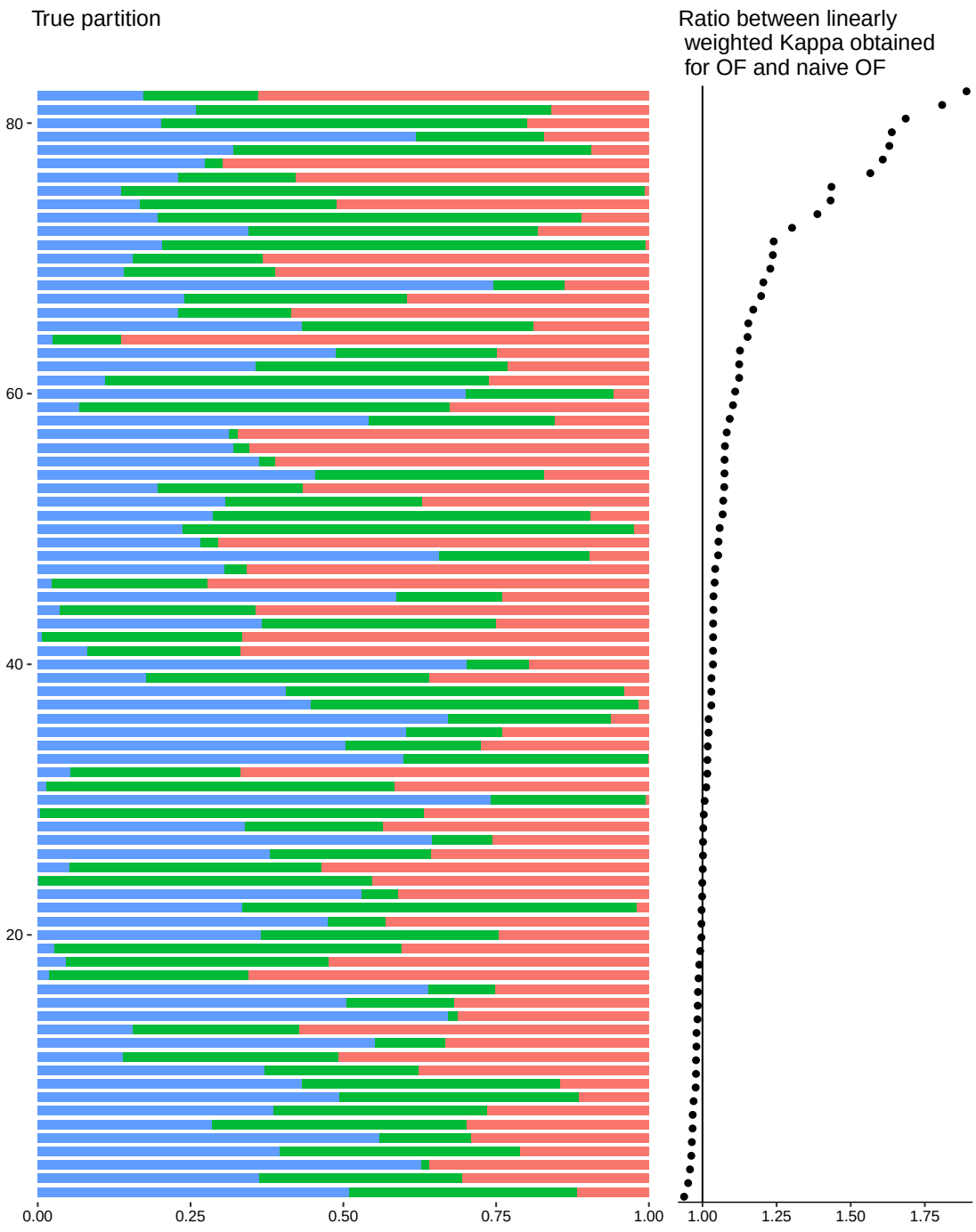
For the settings with "nclass = 9" we again observe the tendency that OF seems to perform especially strong in comparison to naive OF if the widths of the classes around the center of the partition are larger than those of the low and high classes.



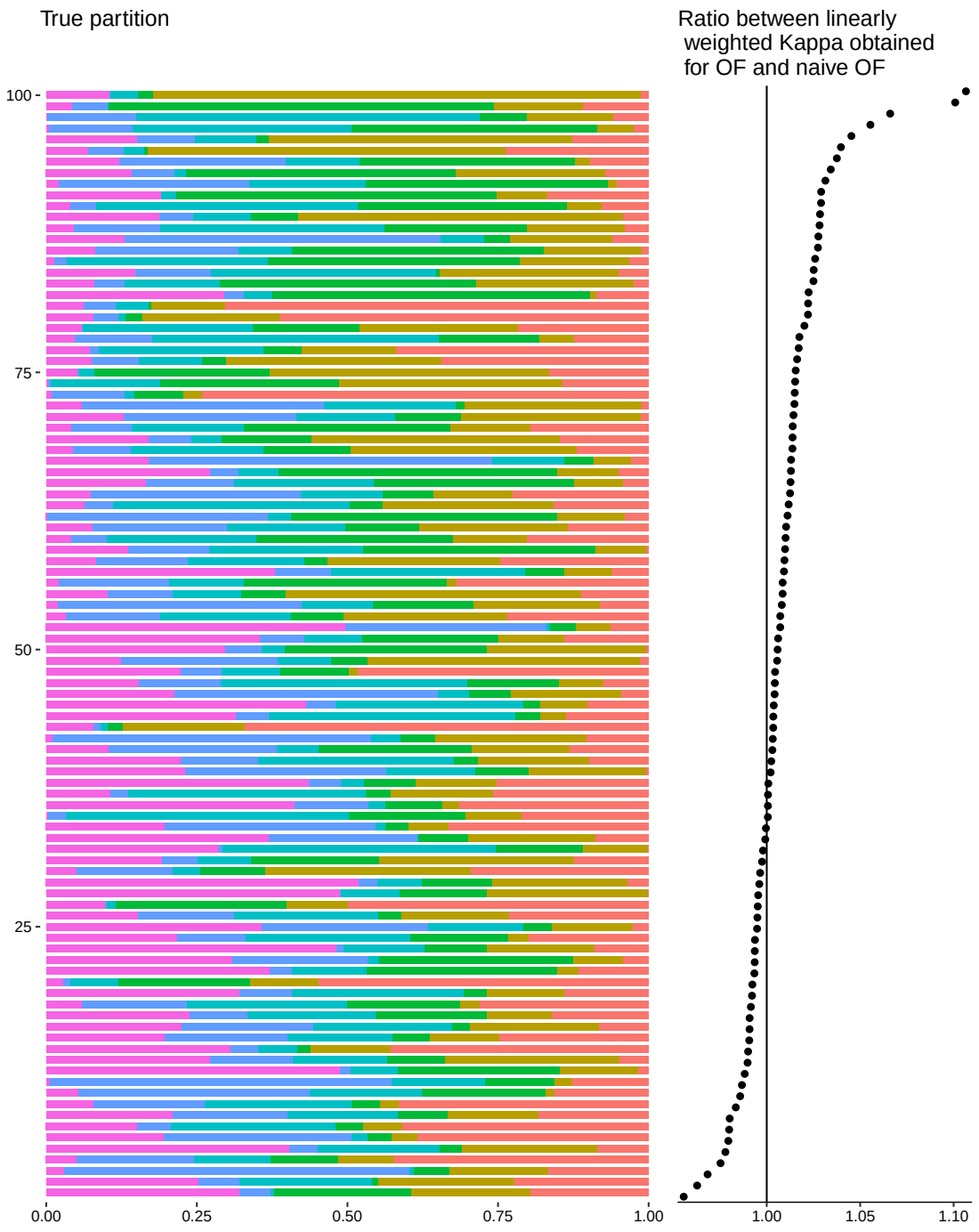
Supplementary Figure 30: True partitions and corresponding performance of OF compared against naive OF - covariates and sample size scenario "correlated_n400" with "nclass = 3". Each row in the plot shows the true partition underlying one of the simulated datasets (left panel) and the ratio between the value of the linearly weighted Kappa obtained for that dataset using OF and using naive OF, respectively (right panel). The rows are ordered in decreasing order with respect to the ratio between the values of the linearly weighted Kappa obtained for the two methods. That is, the partitions for which the performance of OF was best compared to naive OF are in the top part of the figure.



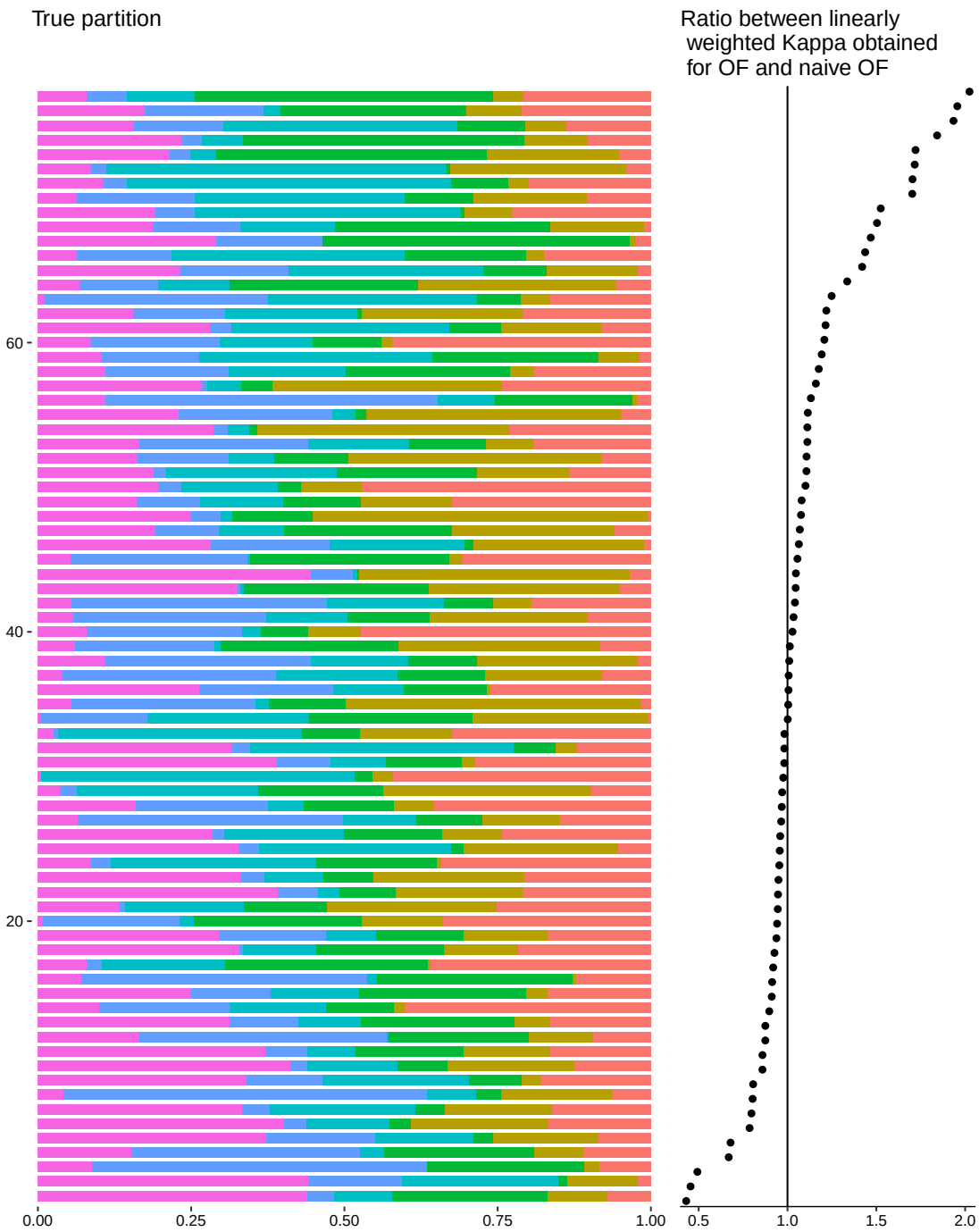
Supplementary Figure 31: True partitions and corresponding performance of OF compared against naive OF - covariates and sample size scenario "independent_n400" with "nclass = 3". Each row in the plot shows the true partition underlying one of the simulated datasets (left panel) and the ratio between the value of the linearly weighted Kappa obtained for that dataset using OF and using naive OF, respectively (right panel). The rows are ordered in decreasing order with respect to the ratio between the values of the linearly weighted Kappa obtained for the two methods. That is, the partitions for which the performance of OF was best compared to naive OF are in the top part of the figure.



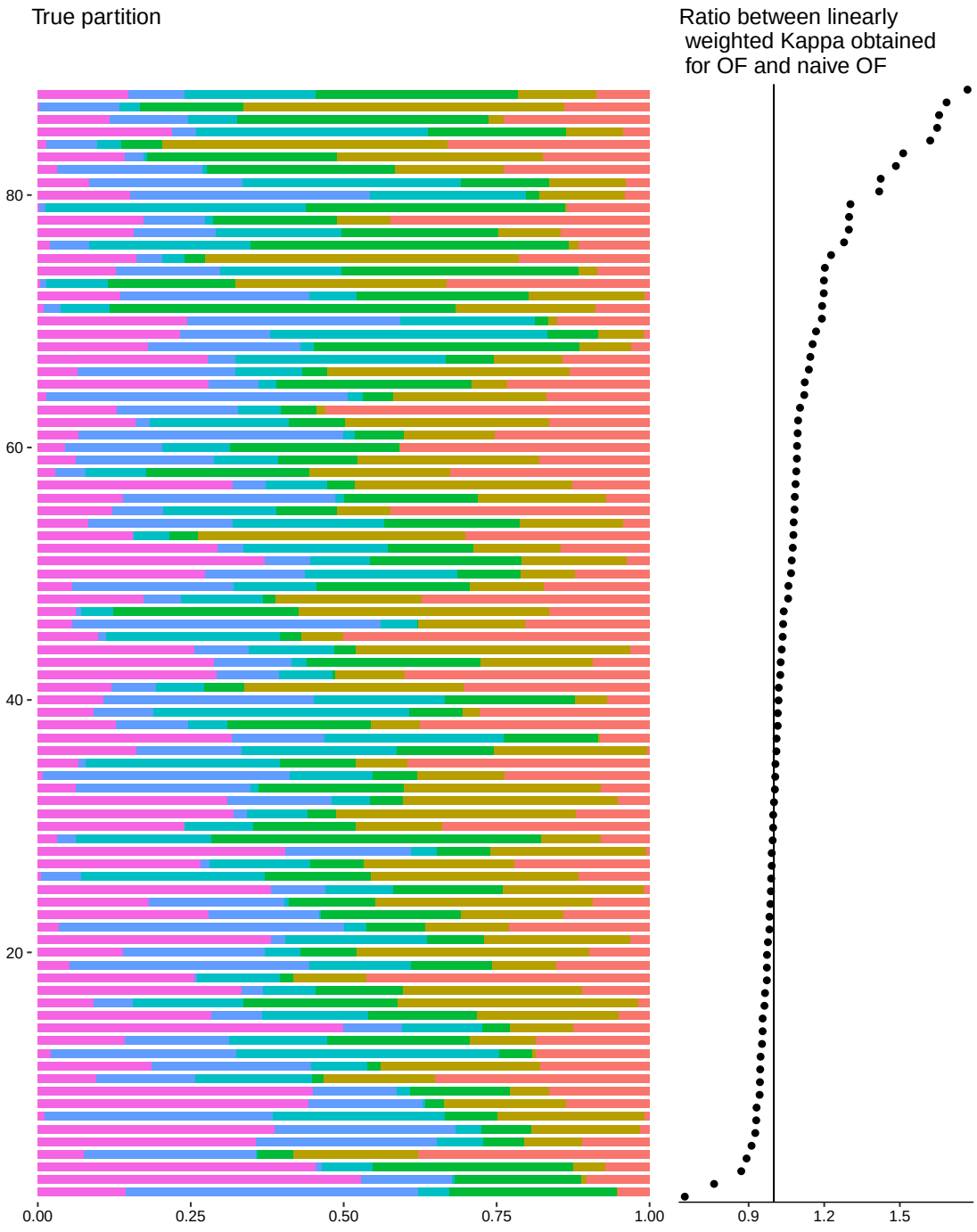
Supplementary Figure 32: True partitions and corresponding performance of OF compared against naive OF - covariates and sample size scenario "highdim" with "nclass = 3". Each row in the plot shows the true partition underlying one of the simulated datasets (left panel) and the ratio between the value of the linearly weighted Kappa obtained for that dataset using OF and using naive OF, respectively (right panel). The rows are ordered in decreasing order with respect to the ratio between the values of the linearly weighted Kappa obtained for the two methods. That is, the partitions for which the performance of OF was best compared to naive OF are in the top part of the figure.



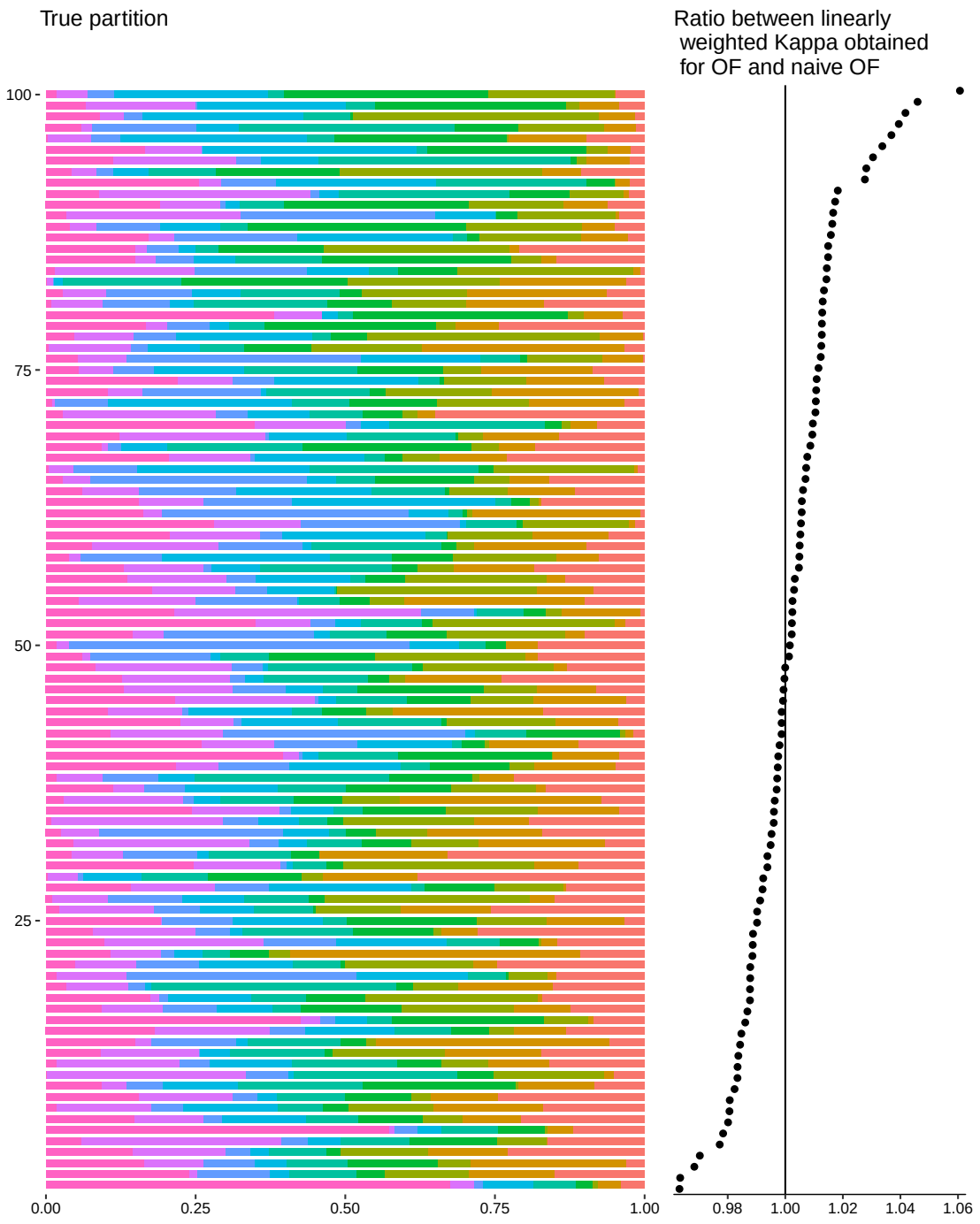
Supplementary Figure 33: True partitions and corresponding performance of OF compared against naive OF - covariates and sample size scenario "correlated_n400" with "nclass = 6". Each row in the plot shows the true partition underlying one of the simulated datasets (left panel) and the ratio between the value of the linearly weighted Kappa obtained for that dataset using OF and using naive OF, respectively (right panel). The rows are ordered in decreasing order with respect to the ratio between the values of the linearly weighted Kappa obtained for the two methods. That is, the partitions for which the performance of OF was best compared to naive OF are in the top part of the figure.



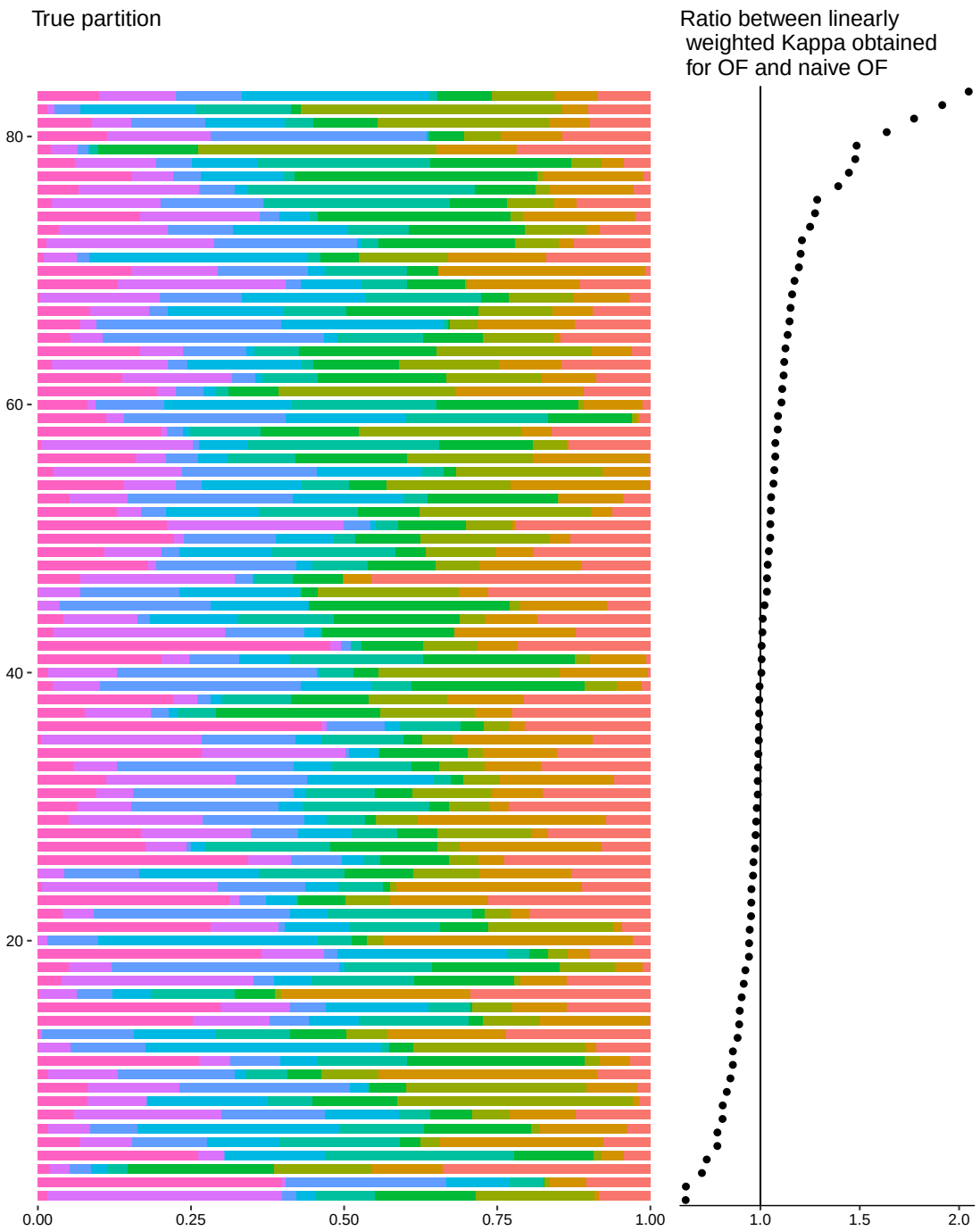
Supplementary Figure 34: True partitions and corresponding performance of OF compared against naive OF - covariates and sample size scenario "independent_n400" with "nclass = 6". Each row in the plot shows the true partition underlying one of the simulated datasets (left panel) and the ratio between the value of the linearly weighted Kappa obtained for that dataset using OF and using naive OF, respectively (right panel). The rows are ordered in decreasing order with respect to the ratio between the values of the linearly weighted Kappa obtained for the two methods. That is, the partitions for which the performance of OF was best compared to naive OF are in the top part of the figure.



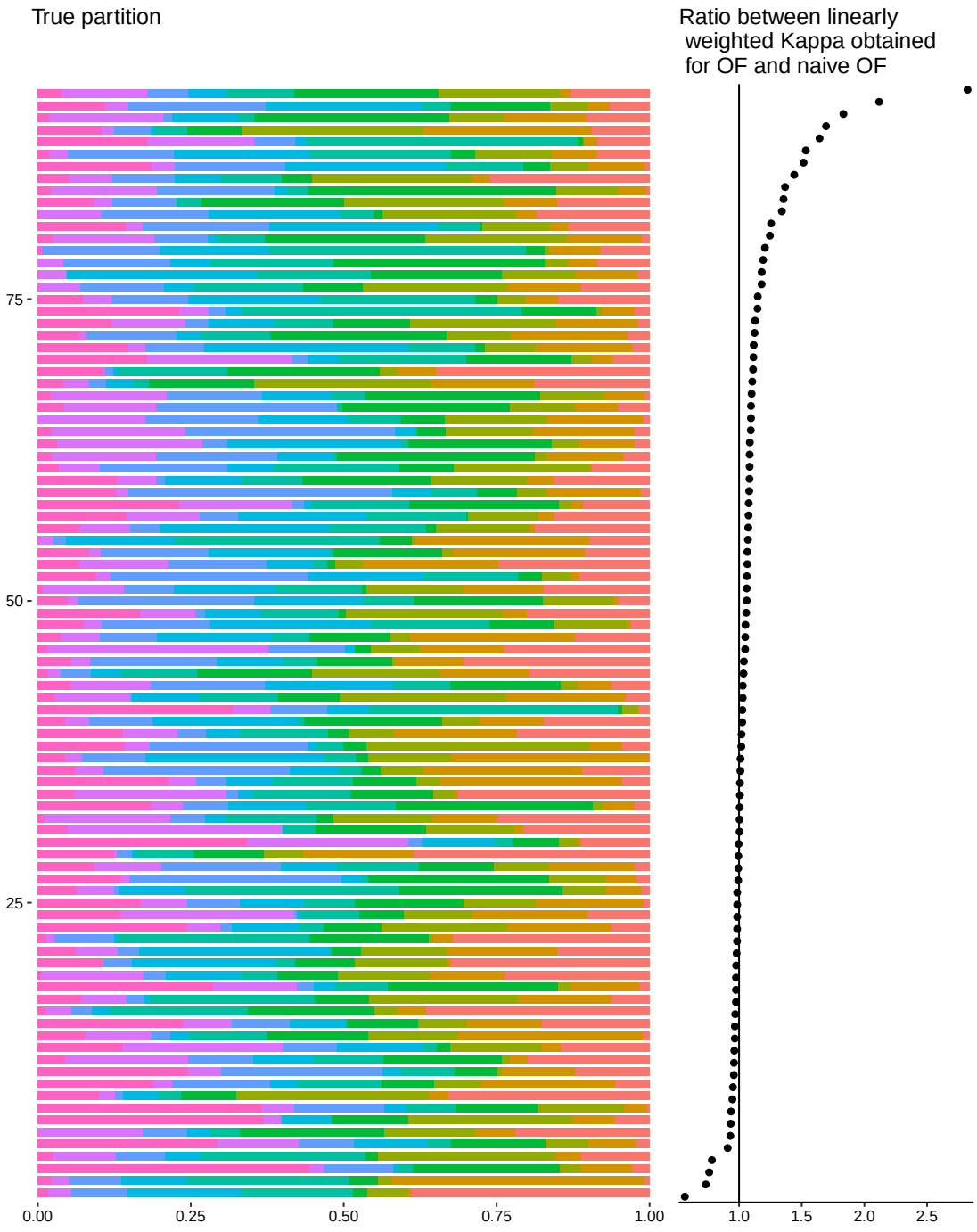
Supplementary Figure 35: True partitions and corresponding performance of OF compared against naive OF - covariates and sample size scenario "highdim" with "nclass = 6". Each row in the plot shows the true partition underlying one of the simulated datasets (left panel) and the ratio between the value of the linearly weighted Kappa obtained for that dataset using OF and using naive OF, respectively (right panel). The rows are ordered in decreasing order with respect to the ratio between the values of the linearly weighted Kappa obtained for the two methods. That is, the partitions for which the performance of OF was best compared to naive OF are in the top part of the figure.



Supplementary Figure 36: True partitions and corresponding performance of OF compared against naive OF - covariates and sample size scenario "correlated_n400" with "nclass = 9". Each row in the plot shows the true partition underlying one of the simulated datasets (left panel) and the ratio between the value of the linearly weighted Kappa obtained for that dataset using OF and using naive OF, respectively (right panel). The rows are ordered in decreasing order with respect to the ratio between the values of the linearly weighted Kappa obtained for the two methods. That is, the partitions for which the performance of OF was best compared to naive OF are in the top part of the figure.



Supplementary Figure 37: True partitions and corresponding performance of OF compared against naive OF - covariates and sample size scenario "independent_n400" with "nclass = 9". Each row in the plot shows the true partition underlying one of the simulated datasets (left panel) and the ratio between the value of the linearly weighted Kappa obtained for that dataset using OF and using naive OF, respectively (right panel). The rows are ordered in decreasing order with respect to the ratio between the values of the linearly weighted Kappa obtained for the two methods. That is, the partitions for which the performance of OF was best compared to naive OF are in the top part of the figure.



Supplementary Figure 38: True partitions and corresponding performance of OF compared against naive OF - covariates and sample size scenario "highdim" with "nclass = 9". Each row in the plot shows the true partition underlying one of the simulated datasets (left panel) and the ratio between the value of the linearly weighted Kappa obtained for that dataset using OF and using naive OF, respectively (right panel). The rows are ordered in decreasing order with respect to the ratio between the values of the linearly weighted Kappa obtained for the two methods. That is, the partitions for which the performance of OF was best compared to naive OF are in the top part of the figure.

G Influence of the choice of the performance function used in the OF algorithm

In the analyses presented in the main paper and in the other sections of the Supplementary Material, the variant g_{clequal} of the performance function was used. This variant of the performance function attributes the same weight to the Youden's index of each class. It thus should lead to a fairly balanced classification performance across the classes. The performance function g_{clprop} by contrast weighs the Youden's indices proportional to the class sizes. This should have the effect that observations from large classes are classified with particular accuracy, which in turn should have the effect that more observations in total are correctly classified, at the expense, however, of a lower performance with respect to classifying correctly observations from the small classes. The last of the considered special cases of the performance functions is g_{clj}. This version attributes full weight to the Youden's index of class j, assigning zero weight to the Youden's indices of the other classes. Therefore, this performance function should lead to a strong prediction performance with respect to distinguishing observations in class j from observations not in class j.

In the analysis presented in this section, how far these three different variants of the performance function are actually associated with the specific kinds of prediction performance they are intended for was investigated. Again the simulation design presented in section 3.2.1 was used and 50 pairs of training and test datasets per setting were generated. To each training dataset OF was applied using the performance functions g_{clequal}, g_{clprop}, and g_{clj} with $j = 2$ and the performance was evaluated on the test dataset using the performance metrics described below.

In the following TP_j ($j \in \{1, \ldots, J\}$) denotes the number of observations in a test dataset which are in class j and for which class j is predicted. Moreover, FN_j denotes the number of test observations in class j for which a different class than class j is predicted. The sensitivity for the prediction of class j is thus: $TPR_j = TP_j/(FN_j + TP_j)$.

Two common ways of averaging class-specific sensitivities across classes are the macro-average and the micro-average [Antonie and Zaïane, 2002], termed MaA and MiA, respectively, in the following. MaA and MiA are given by:

$$\text{MaA} = \frac{1}{J}\sum_{j=1}^{J}\frac{TP_j}{FN_j + TP_j} = \frac{1}{J}\sum_{j=1}^{J} TPR_j \qquad \text{MiA} = \frac{\sum_{j=1}^{J} TP_j}{\sum_{j=1}^{J} FN_j + TP_j}$$

MaA weighs all classes equally, independent of their sizes [Antonie and Zaïane, 2002]. Thus, among the three performance functions considered, g_{clequal} should lead to optimal MaA values. MiA by contrast weighs all test observations equally, thus favoring larger classes [Antonie and Zaïane, 2002]. Therefore, g_{clprop} should lead to optimal MiA values among the three performance functions. Finally, g_{clj} with $j = 2$ should lead to optimal values of TPR_2.

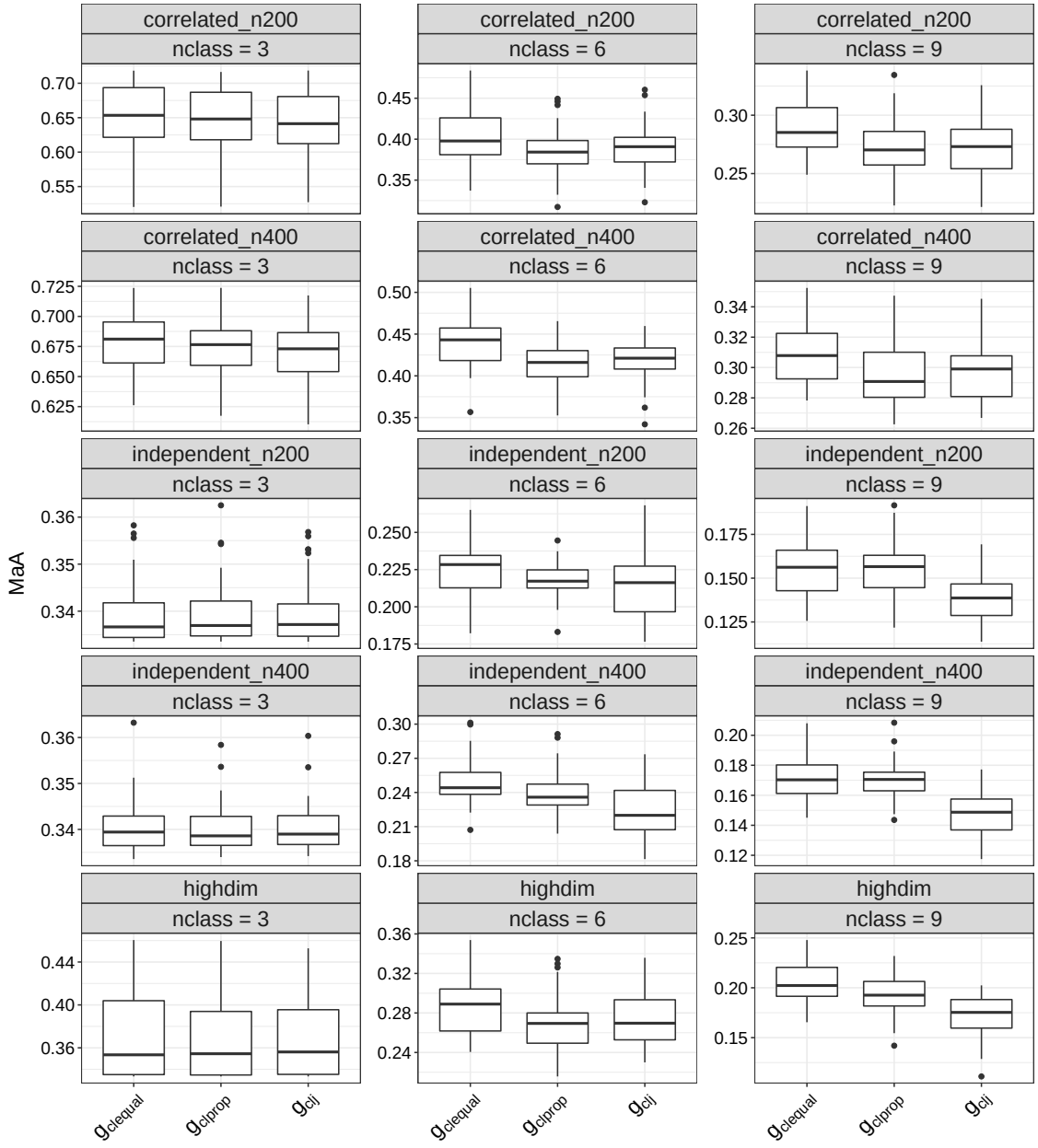
Supplementary Figures 39 to 44 show the values of MaA, MiA, and TPR_2 obtained for all simulation settings.

For some settings g_{clequal} clearly delivered the highest MaA values, while for other settings there is only a slight or no improvement over the other variants of the performance functions considered (Supplementary Figures 39 and 40). For the settings with equal class widths, there are stronger differences between the MaA values obtained for different performance functions. The

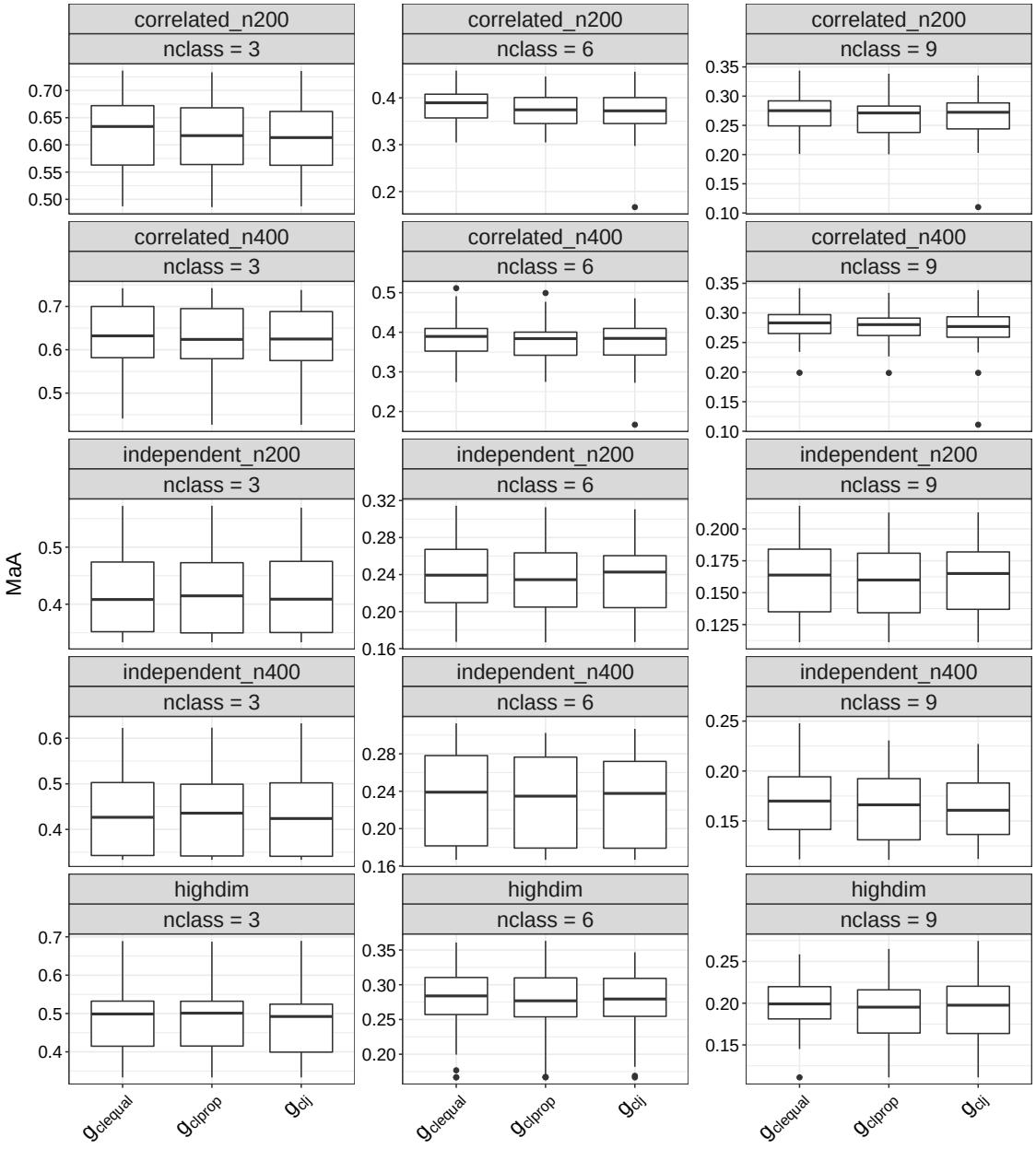
variant g_{clequal} tends to outperform the other two variants of the performance function stronger for "nclass = 6" and "nclass = 9" than for "nclass = 3". This suggests that using g_{clequal} is more effective in cases of higher numbers of classes.

In the case of the MiA values, the picture is similar (Supplementary Figures 41 and 42). For some settings, we observe slightly higher MiA values for g_{clprop} than for g_{clequal}, while for other settings there is no difference between these two functions. In general, the differences are, however, much smaller than in the case of the MaA values. The performance function g_{clj} with $j = 2$ performed considerably worse for some settings and did not perform better than the other two functions in any setting.

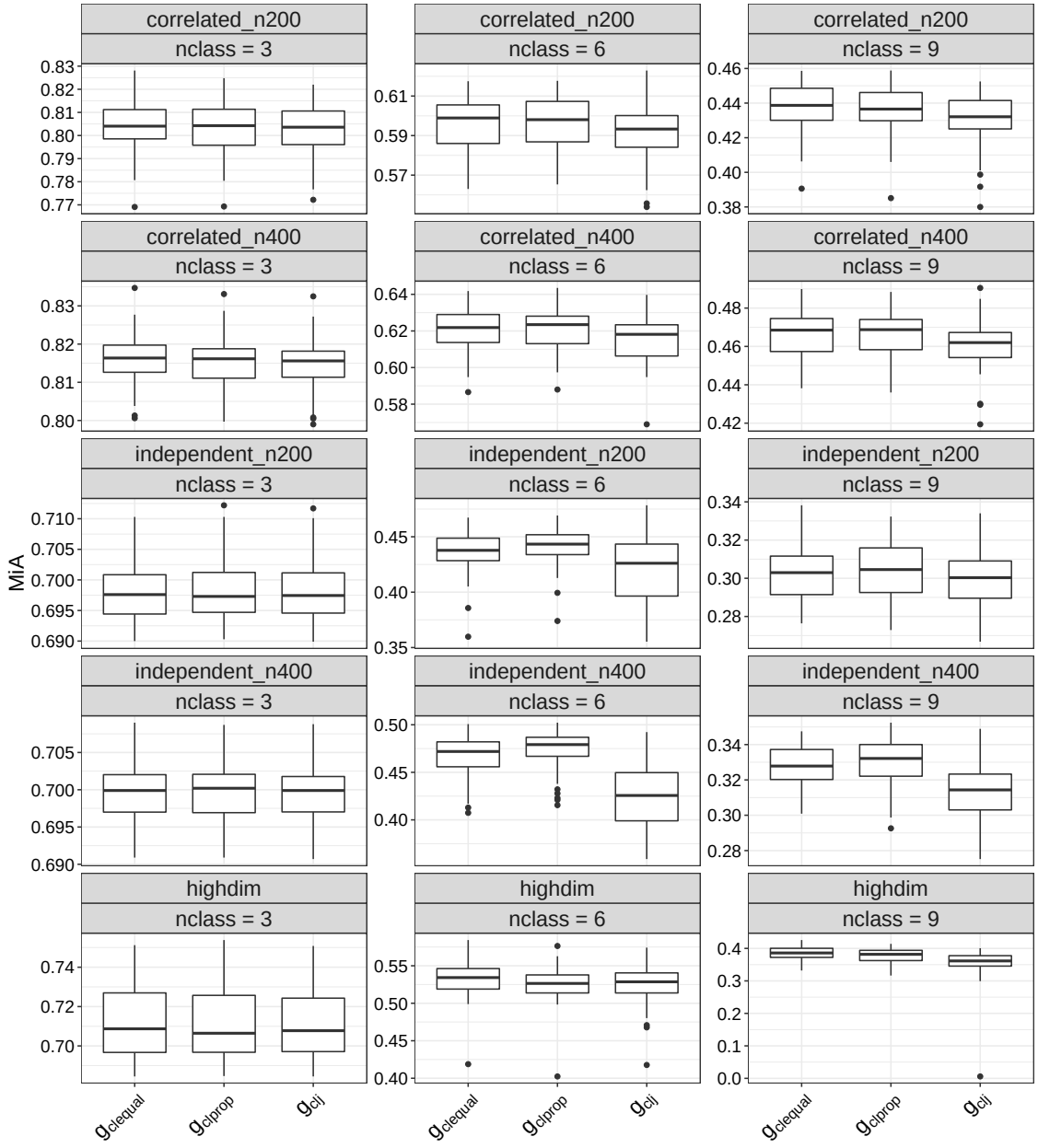
For most settings, g_{clj} with $j = 2$ is clearly better than the other two functions with respect to the TPR_2 values (Supplementary Figures 43 and 44). Again, the improvement over the other two functions is higher for "nclass = 6" and "nclass = 9" than for "nclass = 3". This is probably explainable by the fact that for "nclass = 3" the class 2 that is considered by g_{clj} with $j = 2$ is the class in the middle: Classes in the middle of the class value range are predicted more frequently than classes on the margins of the class value range, which is why putting the focus on class 2 by using the performance function g_{clj} with $j = 2$ might be less effective in the case of "nclass = 3". For "nclass = 6" and "nclass = 9", class 2 is, by contrast, closer to the lower margin of the class range, which is why for these settings the OF algorithm probably profits more from the priority attributed to class 2 through the use of g_{clj} with $j = 2$ than for "nclass = 3". For the settings with equal class widths we observe for "nclass = 3" TPR_2 values close to one in all settings. This is explainable by the high number of observations of class 2 in these settings and, in particular for the scenarios "independent_n200", "independent_n400", and "highdim", by the fact that class 2 is almost always predicted for "nclass = 3". Another peculiarity we observe for the settings with equal class widths is that for "independent_n200", "independent_n400", and "highdim" the TPR_2 values are close to zero for "nclass = 9". This results from, both, the fact that, as discussed in section E.2, the classes on the margins of the class value range are better predictable in the settings with correlated covariates and by the fact that for "nclass = 9" there are only very few observations in class 2 (in the mean there are 7 and 14 observations in this class for $n = 200$ and $n = 400$, respectively, as a simple simple calculation shows).



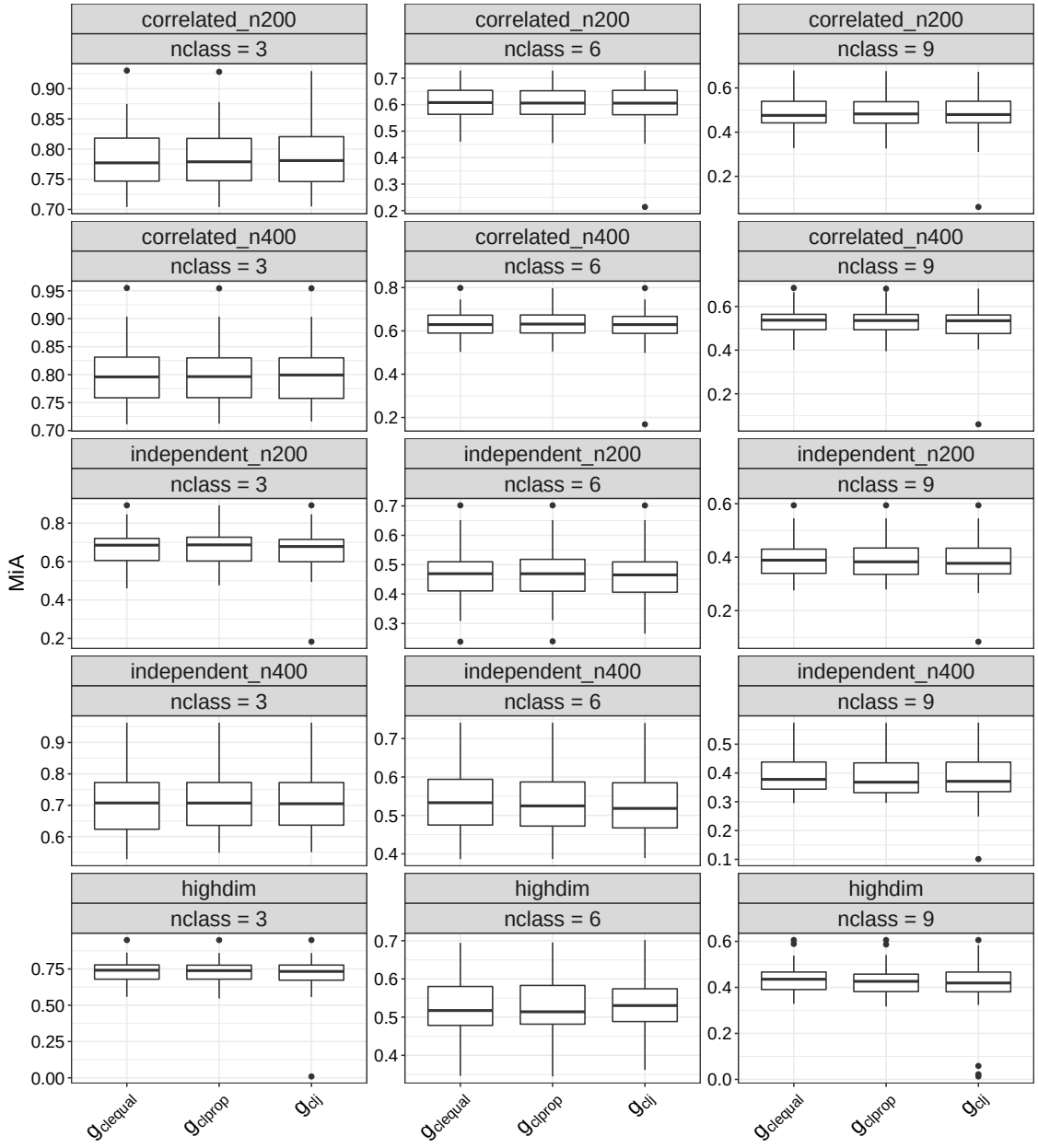
Supplementary Figure 39: MaA values for each simulation setting with equal class widths and each of three performance functions considered. Each boxplot shows the values obtained on the corresponding test dataset for each of the 50 simulation iterations.



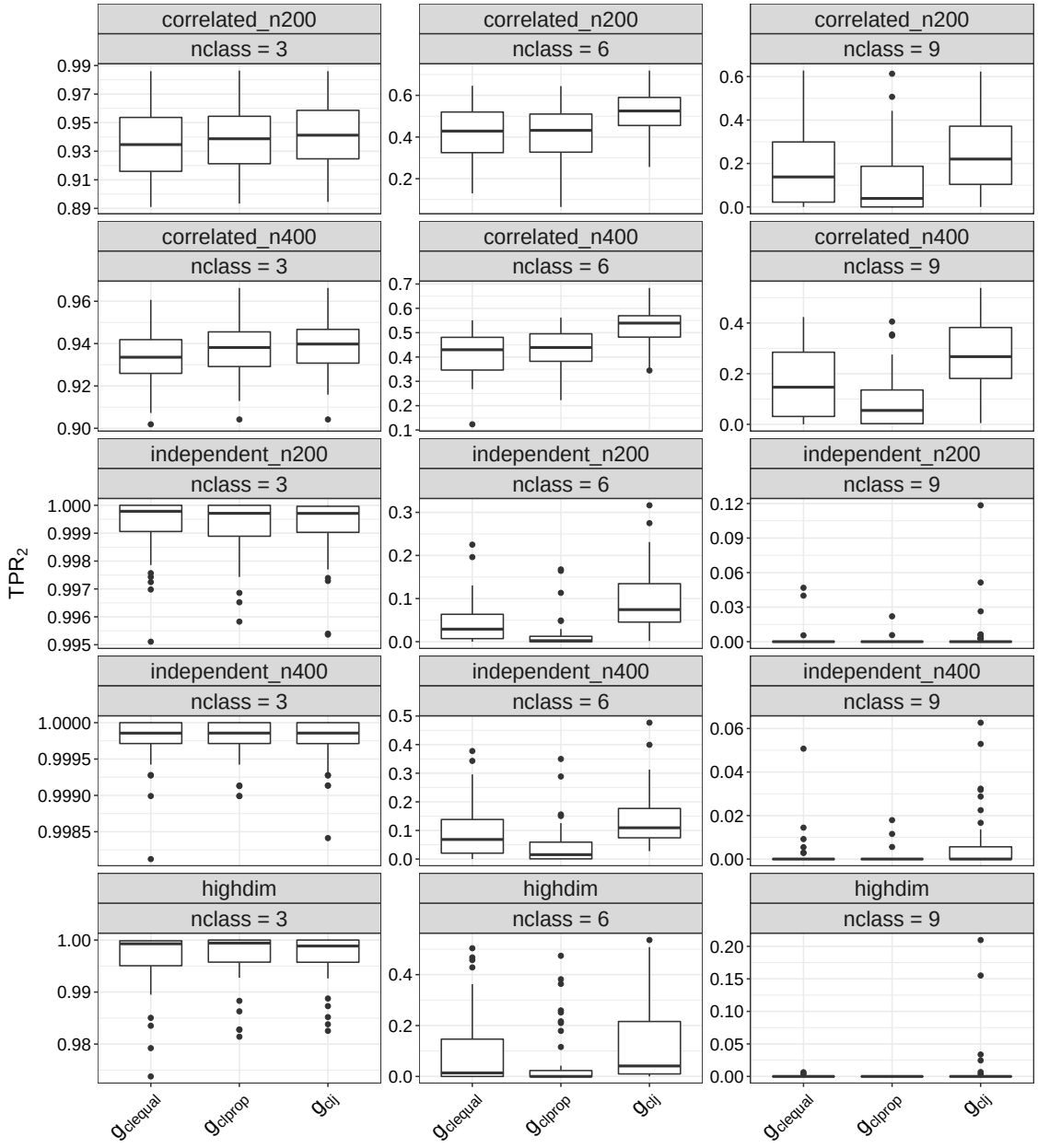
Supplementary Figure 40: MaA values for each simulation setting with random class widths and each of three performance functions considered. Each boxplot shows the values obtained on the corresponding test dataset for each of the 50 simulation iterations.



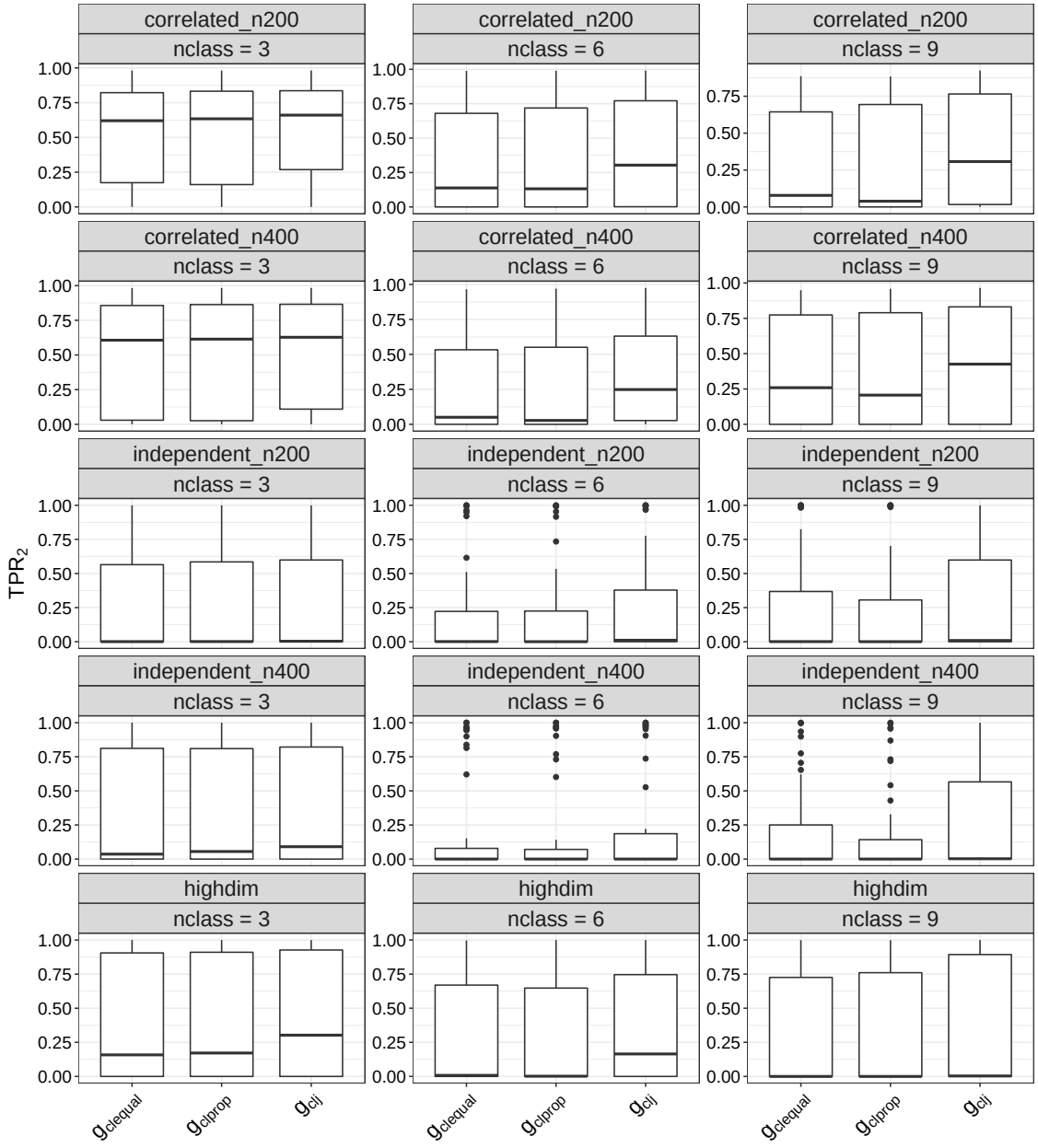
Supplementary Figure 41: MiA values for each simulation setting with equal class widths and each of three performance functions considered. Each boxplot shows the values obtained on the corresponding test dataset for each of the 50 simulation iterations.



Supplementary Figure 42: MiA values for each simulation setting with random class widths and each of three performance functions considered. Each boxplot shows the values obtained on the corresponding test dataset for each of the 50 simulation iterations.



Supplementary Figure 43: TPR_2 values for each simulation setting with equal class widths and each of three performance functions considered. Each boxplot shows the values obtained on the corresponding test dataset for each of the 50 simulation iterations.



Supplementary Figure 44: TPR_2 values for each simulation setting with random class widths and each of three performance functions considered. Each boxplot shows the values obtained on the corresponding test dataset for each of the 50 simulation iterations.

H Hyperparameters in the OF algorithm: appropriateness of their default values and robustness of the results with respect to the choices of their values

H.1 Heuristic discussion on the influences of the hyperparameter values on the performance of OF

The larger the value of B_{sets} is chosen, the nearer the B_{bestsets} best score sets will be to the optimal score set, that is, the score set that is associated with the greatest OOB prediction performance. Moreover, the larger the value of $B_{\text{ntreeprior}}$, the more reliable will be the sc_b values. A large value is also better in the case of B_{ntree}, because the prediction performance and the stability of the VI ranking will be higher for a higher number of trees. While the values of B_{sets}, $B_{\text{ntreeprior}}$, and B_{ntree} should thus in principle be chosen as large as possible, extremely large values of these parameters would lead to a too high computational burden. However, it is not necessary to choose such extremely large values, because it is sufficient to set these parameters to values that are large enough for practical purposes. The latter means that the values have to lie in an order of magnitude for which no substantial improvement of the results would be attained by increasing these values further.

The larger N_{perm} is chosen, the more different the class width rankings of directly consecutive sets $\{d_{b-1,1},\ldots,d_{b-1,J+1}\}$ and $\{d_{b,1},\ldots,d_{b,J+1}\}$ ($b \in \{2,\ldots,B_{\text{sets}}\}$) will be from one another. However, the more different the class width ranking from iteration b is to that from iteration $b-1$, the more similar it tends to be to the class width ranking from iteration $b-2$. As an illustration: in the extreme case $N_{\text{perm}} \to \infty$, there will merely be two different sets in the collection of sets $\{d_{b,1},\ldots,d_{b,J+1}\}$ ($b \in \{1,\ldots,B_{\text{sets}}\}$), because $\{d_{b-2,1},\ldots,d_{b-2,J+1}\}$ will be the same as $\{d_{b,1},\ldots,d_{b,J+1}\}$ for $b \in \{3,\ldots,B_{\text{sets}}\}$. Leaving aside this extreme case, a large value of N_{perm} has the effect that the resulting collection of class width rankings can be broken down into pairs of clusters that feature very similar class width rankings within each of the two clusters, but very dissimilar class width rankings across the two clusters. In the case of $N_{\text{perm}} = 1$, that is, if not choosing the class width ranking of iteration b distant from that of iteration $b-1$, the probability of choosing very dissimilar sets of class width rankings would naturally be smaller. For a large number J of classes it it is, however, important to choose such sets of strongly dissimilar class width rankings, which will be explained in the following. In the case of a large J, the optimal class width set cannot be approximated as precisely as in the case of a small J. In this situation, the performance of OF will mainly depend on whether or not the optimized class width set features characteristic patterns of the optimal class width set, for example, a very large class width for class 1 and a particularly small class width for class 3. For a large N_{perm}, that is, when using a collection of class width rankings that features pairs of clusters of similar rankings, where these clusters are disparate from each other, we are likely to generate characteristic patterns of the optimal class width set that are associated with a strong performance. Staying with the above example, in such a collection of class width rankings there are many class width rankings for which class 1 has highest rank and at the same time class 3 has lowest rank. In this situation, it is likely that for some of the class width sets considered, class 1 features a very large width and class 3

a very small width. That is, the desired characteristic pattern of the class width ranking that is associated with a strong performance of the OF will likely occur in several of the generated class width sets. Put shortly, choosing a high value for N_{perm} leads to generating large tuples of class width sets with distinct characteristics of the optimal class width set, which are associated with a strong performance of the OF if taken into account. Note that for small numbers of classes (more precisely for $J! < B_{\text{sets}}$) a different algorithm is considered in the R package `ordinalForest` that uses all $J!$ possible class width rankings and thus does not involve N_{perm} (see section A).

The optimal value of the number B_{bestsets} of score sets used to calculate the optimized score set depends on the number B_{sets} of score sets tried. If B_{sets} is relatively small, a value of B_{bestsets} that is too large will lead to the inclusion of suboptimal score sets too distinct from the optimal score set, which is why in this situation the optimized score set will be far from the optimal score set. If, again for a relatively small B_{sets}, the value of B_{bestsets} is, by contrast, too small, the variance of the optimized score set will be too high. For larger values of B_{sets}, the results are less sensitive to choosing a relatively large value of B_{bestsets}, because for a large B_{sets} the number of tried score sets that are close to the optimal score set is higher. Moreover, the variance of the best score sets tried, that is, those with the highest sc_b values, will be smaller for a large B_{sets}, because by trying a larger number of score sets the best score sets will be nearer to the optimal score set. Therefore the results are also less sensitive to choosing a relatively small value of B_{bestsets} when B_{sets} is large. Summarizing, an adequate choice of B_{bestsets} is particularly important when B_{sets} is relatively small, where the choice of the value of B_{bestsets} depends on the value of B_{sets}.

H.2 Simulation study on the appropriateness of the default hyperparameter values and the robustness of the OF performance with respect to the choices of the values of the hyperparameters

Above the forms of the influences of the different hyperparameters of the OF algorithm on its performance were discussed. In the following, an empirical analysis on the sensitivity of the performance of OF to varying the values of the hyperparameters will be presented. Of particular interest will be to assess whether the default values considered for the hyperparameters are appropriate.

In this analysis, for two simulations settings, namely (i) "correlated_n200" with equal class widths and "nclass = 6" and (ii) "highdim" with equal class widths and "nclass = 6", twenty pairs of training and test datasets were generated. OF was applied to each training dataset with different values of the various hyperparameters and the values of the linearly weighted Kappa were calculated on each corresponding test dataset. The simulation design was kept simple for reasons of clarity and because the main interests of this study were merely to ensure that the chosen default values of the hyperparameters are not grossly misspecified and to exclude the possibility that there is a high sensitivity of the results regarding the choices of the values of these hyperparameters. Equal class widths were considered, because for this scenario the results are less variable than for random class widths. The latter makes it easier to study trends in the Kappa values in dependency of the hyperparameter values. Moreover, one of the two settings features a low-dimensional covariate vector and the other a high-dimensional covariate vector. For the setting with low-dimensional covariate vector the scenario with correlated covariates was chosen,

because for this scenario the values of the linearly weighted Kappa are higher. Therefore, we can expect clearer trends in the Kappa values in dependency of the hyperparameter values. Finally, the scenario with a smaller sample size was chosen for the scenario with correlated covariates, because naturally the dependency of the results on the choices of the hyperparameter values can be expected to be stronger for smaller sample sizes.

The following values were considered for the different hyperparameters:

- B_{sets} and B_{bestsets}: all possible combinations of $B_{\text{sets}} \in \{500, 1000, 5000, 10000\}$ and $B_{\text{bestsets}} \in \{5, 10, 100, 1000\}$ except for $B_{\text{sets}} = 500$ with $B_{\text{bestsets}} = 1000$ and $B_{\text{sets}} = 1000$ with $B_{\text{bestsets}} = 1000$
- $B_{\text{ntreeprior}}$: 10, 50, 100, 500, 1000
- B_{ntree}: 500, 1000, 5000, 10000
- N_{perm}: 1, 100, 500, 1000

While varying the values of each hyperparameter all other hyperparameters were set to their default values.

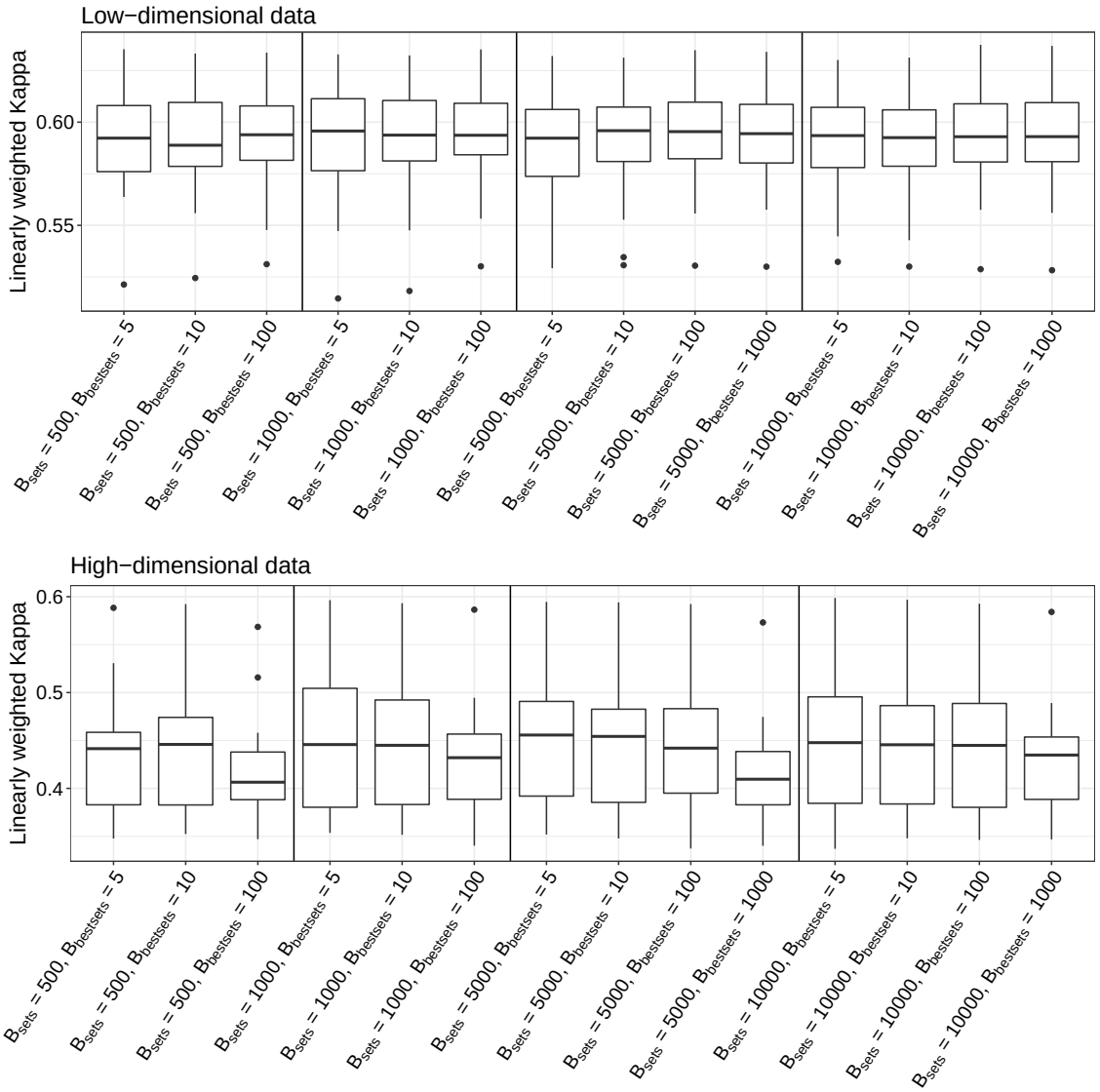
Supplementary Figures 45 and 46 show the results. In many cases the boxplots hardly vary across the different values of the hyperparameters. The influence of the combination of B_{sets} and B_{bestsets} and that of $B_{\text{ntreeprior}}$ is generally stronger than the influences of B_{ntree} and N_{perm}.

For the combination of B_{sets} and B_{bestsets} (Supplementary Figure 45) the differences across the results obtained for the various combinations of parameter values are stronger for the high-dimensional setting than for the low-dimensional setting. For the latter, the only notable observation to be made is that for the smallest number of B_{bestsets} (i.e., $B_{\text{bestsets}} = 5$), the performance tends to be slightly worse. In the case of the high-dimensional setting, the performance is clearly worse for very high values of B_{bestsets}. For $B_{\text{sets}} \in \{5000, 10000\}$ this deterioration is only seen for $B_{\text{bestsets}} = 1000$ not for $B_{\text{bestsets}} = 100$, while for $B_{\text{sets}} \in \{500, 1000\}$ the performance clearly declines for $B_{\text{bestsets}} = 100$. This illustrates that for a large value of B_{sets} the results are less sensitive to setting the value of B_{bestsets} overly large. For the high-dimensional setting the performance is slightly better for $B_{\text{sets}} = 1000$ than for $B_{\text{sets}} = 500$, but there is no notable gain in performance for higher values of B_{sets} than 1000. In both settings the combination of the default values $B_{\text{sets}} = 1000$ and $B_{\text{bestsets}} = 10$ works well.

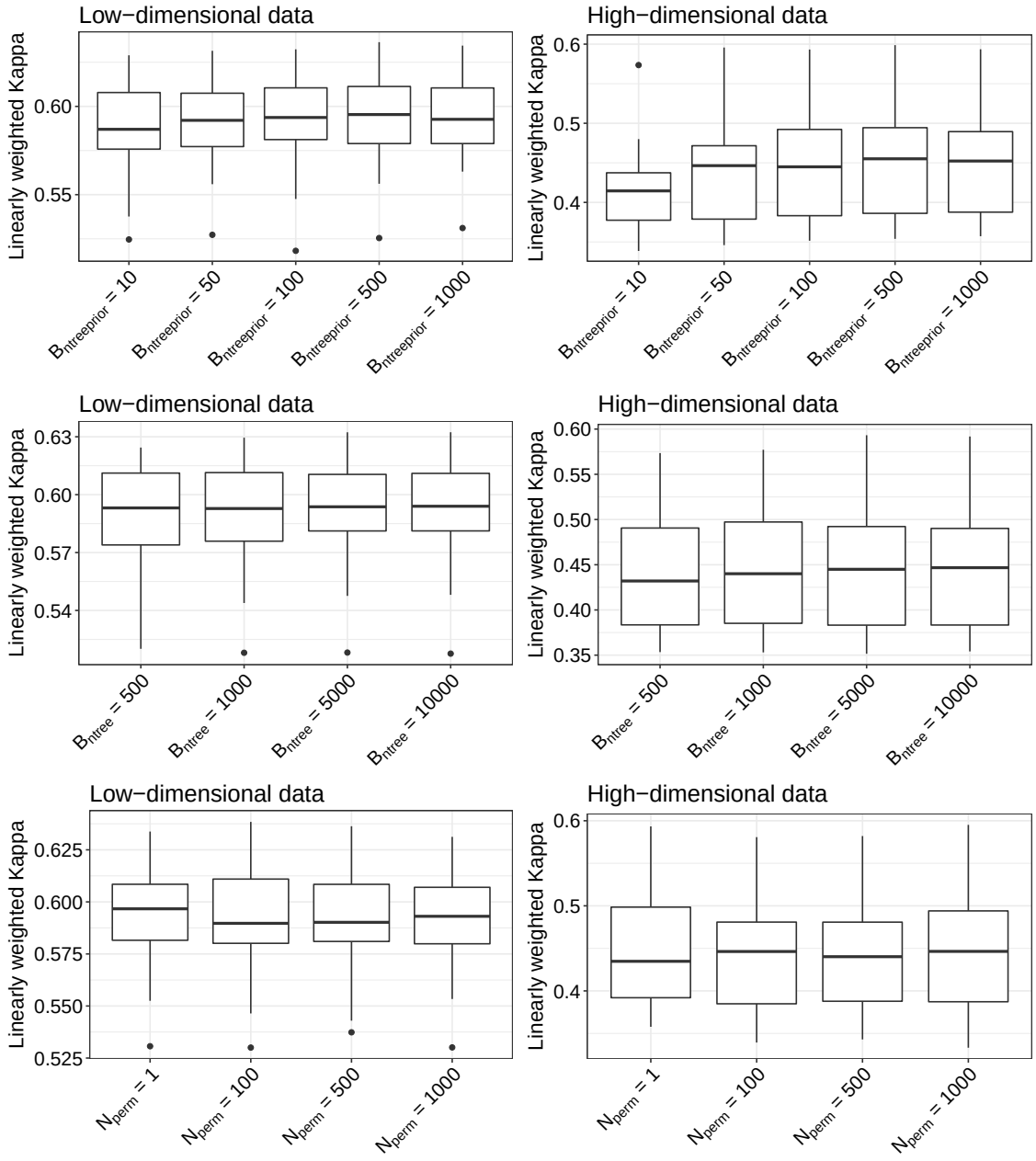
For both settings the performance is better for higher values of $B_{\text{ntreeprior}}$, where there is no notable further improvement for values higher than $B_{\text{ntreeprior}} = 100$ (Supplementary Figure 46). However, the influence of $B_{\text{ntreeprior}}$ is considerably stronger for the high-dimensional setting. This can probably be explained by the fact that there is only a small percentage of influential variables in the high-dimensional setting, which might not be selected frequently enough for a small number of trees. Summarizing, for both settings the default value 100 for $B_{\text{ntreeprior}}$ was sufficiently high. Nevertheless, for ultra-high-dimensional datasets $B_{\text{ntreeprior}}$ might have to be set even larger than 100 to warrant a sufficient accuracy of the sc_b values.

In the case of B_{ntree} for both settings there are no notable differences between the Kappa values obtained for 500, the lowest B_{ntree} value considered and for 10000, the highest B_{ntree} value considered. Therefore, the default value $B_{\text{ntree}} = 5000$ seems sufficient.

The influence of the value of N_{perm} is marginal for both settings. We do not observe a better performance for choosing the value of N_{perm} larger than one. Note however, that for both of the considered settings the number of classes is only six. As $J! = 6! = 720 < 1000 = B_{\text{sets}}$, the version of the algorithm considered for generating the B_{sets} sets of class widths that does not involve N_{perm} (see section A for details) would have been used as the default option in `ordinalForest`. For a higher number of classes, choosing a sufficiently large value of N_{perm} might be more important. Nevertheless, as we do not observe any notable differences between the results obtained for choosing N_{perm} equal to one and that obtained for choosing N_{perm} larger than one, it can be assumed that the value of N_{perm} does not have a strong influence on performance.



Supplementary Figure 45: Values of linearly weighted Kappa obtained for different combinations of B_{sets} values and B_{bestsets} values. Each boxplot shows the values obtained on the corresponding test dataset for each of the 20 iterations of the simulation setting with "correlated_n200", equal class widths, and "nclass = 6" (upper panel) and the simulation setting with "highdim", equal class widths, and "nclass = 6" (lower panel). The values of all other hyperparameters than B_{sets} and B_{bestsets} were set to their default values (see second paragraph of section 3 of the main paper).



Supplementary Figure 46: Values of linearly weighted Kappa obtained for different $B_{\text{ntreeprior}}$ values (upper panels), B_{ntree} values (middle panels), and N_{perm} values (lower panels). Each boxplot shows the values obtained on the corresponding test dataset for each of the 20 iterations of the simulation setting with "correlated_n200", equal class widths, and "nclass = 6" (left panels) and the simulation setting with "highdim", equal class widths, and "nclass = 6" (right panels). In each case, the values of all other hyperparameters than the ones under investigation were set to their default values (see second paragraph of section 3 of the main paper).

References

M-L Antonie and O R Zaïane. Text document categorization by term association. In V Kumar, S Tsumoto, N Zhong, P S Yu, and X Wu, editors, *Proceedings of the 2002 IEEE International Conference on Data Mining*, pages 19–26, 2002.